A gentle transition from Java programming to Web Services using XML-RPC

Marc Conrad - Tim French
Department of Computing and IT
Faculty of Creative Arts and Technology
University of Luton, LU1 3JU, UK
Marc.Conrad@luton.ac.uk, Tim.French@luton.ac.uk

Abstract
Exposing students to leading edge vocational areas of relevance such as Web Services can be difficult. We show a lightweight approach by embedding a key component of Web Services within a Level 3 BSc module in Distributed Computing. We present a ready to use collection of lecture slides and student activities based on XML-RPC. In addition we show that this material addresses the central topics in the context of web services as identified by Draganova (2003).

1. Introduction

The continuing global expansion of various forms of e-commerce as exemplified by B2C (Business to Consumer), B2B (Business to Business) and emergent Web Services, have served to raise the profile of on-line architectures and protocols based upon XML (Extensible markup Language) standards and its many derivative forms, such as the SOAP (Simple Object Access Protocol). For examples of implementations of Web Services see e.g. W3C (2003), Apache (2002-2004), or Sun (2002).

Indeed, the rapid pace of development in this area within industrial settings has created an equally rapid demand upon on the part of University curricula both at Undergraduate and Postgraduate levels to retain industrial credibility. We have sought to respond to this challenge by seeking to embed Web Services (more specifically XML messaging containing embedded Remote Procedure Calls (RPC's)) within our existing curriculum in "bite size" pieces.

That is to say, our aim is to expose students to a small "taster" of this leading edge and vocationally credible area, without seeking to compromise academic rigor or unbalancing their overall programme of study within our undergraduate BSc (Hons.) programmes of study.

Hence, we proceed to offer a description of a small unit of study (two weeks) embedded within an existing module together with our reflections so as to stimulate a wider pedagogic debate concerning how, and in what ways, it is possible for existing degree programmes in Computing and Computer Science to successfully seek to expose students engaged in such programmes of study to rapidly evolving industrial practice.

We describe a teaching unit based on the XML-RPC protocol (UserLand, 1998-2004)

> What is XML-RPC?
>
> It's a spec[ification] and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet.
>
> It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.
>
> *Figure 1: The definition of XML-RPC according to http://www.xmlrpc.com.*

that introduces the central techniques that all Web Services rely upon. The students require as a prerequisite a "basic" knowledge of Java. The units have been delivered in the academic years 2003-4 and 2004-5 as part of a 3rd year BSc course in "Distributed Computing" at the University of Luton. They may also be used in other contexts, e.g. in a lecture on Java programming. Each year approximately 100 students attend the class. Anecdotal evidence shows that students respond positively to the activities provided. Almost all of these students were successful in finishing their first activity. The majority of the students successfully mastered the second activity, usually with help from peers or their tutor. The core of the student experience is a balanced mixture of individual work and group work. The teaching material includes lecture slides and activities for practical sessions. It is available (free of charge) from http://perisic.com/xmlrpc (Conrad, 2003), and based on the Apache XML-RPC implementation (Apache, 2001-2003). All the code involved is open-source and can be tailored to individual situations.

The technical requirements are lightweight and minimal. At the University of Luton we use the J2SE 1.4.2 (Sun, 2004) running under a DOS shell combined with the text editor Edit+. However the activities may be performed on any Java developing environment. The computers of course need to be connected over a network. As the connection is established via port 80 (used by Web Services) there are no firewall problems to be expected.

In the remainder of the paper we firstly describe the contents and the purpose of the lecture slides. In section three that follows we describe typical activities and in section four we evaluate the material presented within the context of the learning targets addressed above. We then go on to look at the relationship between XML-RPC and SOAP. Finally we shortly discuss the student experience of activities.

2. The Lecture Slides

The first set of the lecture slides covers the following topics:

1. The role of XML in XML-RPC using the analogy with natural languages (see Figure 2 on the next page);
2. The difference between a *remote* procedure call as compared to a local procedure call;

3. A detailed discussion of the official definition of XML-RPC as presented on the homepage of XML-RPC (see Figure 1) Here the emphasis is an understanding that a distributed framework needs *both* the specification and the implementation of this specification in programming languages;
4. An extensive presentation of Java code for a Server and Client using the Apache XML-RPC implementation.

The second set contains the XML-RPC specification distributed across 13 slides with suitable annotations. This "real life" example covers addresses the following topics:

1. Example of a HTTP header;
2. The basic structure of an XML document;
3. A review of fundamental data types (int, boolean, string, double, arrays, structs);
4. Fault handling;
5. The copyright emphasising on the two main aspects that make up an open standard, i.e. that the specification must not be changed but that everyone is free to implement the specification.

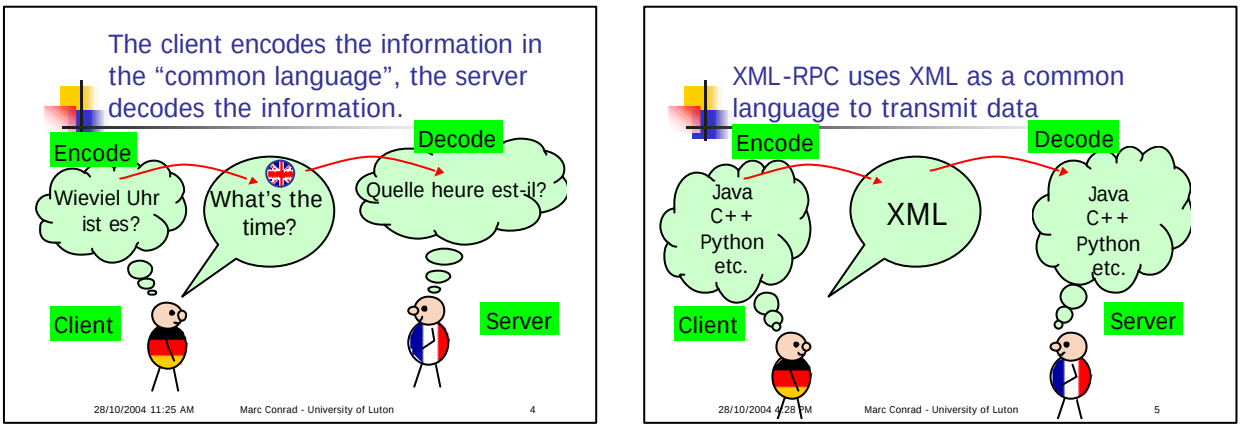


Figure 2: Using an analogy with natural languages

3. Student Activities

The first activity consists of five tasks that can be summarised as follows:

1. Compile and run a server and client program locally (the source code for these programs is provided). The basic web service is that the server on client request delivers a "Hello World" -like string.
2. Modify the code of the server such that a different string is presented (e.g. the name of the student).
3. Distribute the IP number of the machine to other students (for instance the tutor can keep a list of IP numbers with a running server on a whiteboard).
4. Change the code of the client to access a server that is not local.
5. Add a special command into the code that shows the XML that is exchanged between client and server.

Note that this activity does not require significant programming knowledge. An interesting behaviour can be observed at step 3 namely, that students who have been

successful in running their own server motivate their peers to work on their client program, therefore leading to a productive interaction.
The second activity consists of the implementation of a distributed application from the non-distributed Java program shown in Figure 3. This task can be solved using the programs of the first tasks as templates.

4. Impact on Student Understanding of Web Services
Draganova (2003) provided us with a useful template identifying the central topics in the context of web services, namely

- Discovering a location of a service provider;
- Discovering provided services;
- Providing a way of communication with a particular service;
- Providing a way to execute available functions;
- Providing a standard messaging;
- Providing a way of data representation.

```
import javax.swing.*;

public class PrimitiveChat {

// The procedure:
 public String printText(String str) {
    System.out.println( "Received: "+str);
    return "ok";
    }

 public static void main (String [] args) {
  String input;
  PrimitiveChat pc = new PrimitiveChat();

  do {
    input = JOptionPane.showInputDialog("Enter your message");
    if( input != null ) {
        pc.printText(input); // a local procedure call.
        }
    } while ( input != null );
  System.exit(0);
  }
}
```

Figure 3: A non-distributed example program

These topics can be instantiated by using XML-RPC as follows:

Discovering a location of a service provider
Students use two mechanisms to link the application: Firstly, they work on their local machine using the address http://localhost/RPC, then they access a remote location by its IP number. By giving their own IP number to peers to access their server, they are "publishing" a web service. It is then straightforward to question the usefulness of the

IP number and the possibility to replace that by a name (as given in the example in the lecture). This then naturally leads to the discussion of using nameservers etc.

Discovering provided services
In the second activity students are asked to transform a local procedure call into a remote procedure call. When they ask their peers to access this new service they quickly see that it is not only sufficient to hand over the IP number alone, but also the name of the remote procedure, therefore learning the importance of identifying the specification of a service.

Providing a way of communication with a particular service
In the last task of the first activity students examine the XML code that is transmitted between client and server. As XML is human-readable they are able to identify immediately the text inside this code. This experience of course goes hand in hand with the discussion of the XML-RPC specification in the lecture.

Providing a way to execute available functions
A remote procedure call is the simplest way to access a web service. Starting from the well-known knowledge of invoking local methods (procedures) in a Java program the activities and lectures provide a gentle generalisation of that mechanism.

Providing a standard messaging
The target of the second practical is the implementation of a "chat" system at its most basic level, therefore introducing the students to the topic of messaging.

Providing a way of data representation
One of the major advantages of using XML-RPC in the curriculum is that the XML-RPC specification can be covered *in full* in a 90 minute lecture. The data that can be transmitted in an XML-RPC request consists mostly of the usual data types encountered in a programming lecture (Strings, integers, floating points, booleans, arrays) and a very few number of data structures that may be new (structs, base64, and date).

Other topics
In addition to the primary topic of understanding Web Services, by seeing a specification as whole, students also gain a better understanding on "Open Standards".

5. XML-RPC and SOAP

The relationship between XML-RPC and SOAP is best described by Wikipedia (2005): *"[XML-RPC] was first created by Dave Winer of UserLand Software in 1995 with Microsoft. However Microsoft considered it too simple, and started adding functionality. After several rounds of this, the standard was no longer so simple, and became what is now SOAP."* Therefore XML-RPC serves as a good starting point for teaching SOAP. We especially feel that the lightweight nature of the XML-RPC protocol better facilitates the students learning experience in the given time frame of two weeks than an introduction into SOAP directly. As mentioned in section 5 two weeks are enough to cover the *full* picture of XML-RPC including examples and the specification. We see this as a major advantage as for most topics (as HTML, SOAP, XML etc.) a teaching module is able to cover only selected parts of the standard.

6. Student Experience and Feedback

The units have been delivered so far at the University of Luton twice. First in 2003 in a module of 100 students and in 2004 in a module of 160 students. In 2003 there was no formal assessment point associated to the tasks. Any feedback there is based on observation and anecdotal evidence. In 2004 the tasks have been included as part of a student portfolio that is currently (January 2005) reviewed and marked. We will report on these results at the conference and make the details available on the Internet.

The experience so far shows a thoroughly positive response. We made the observation that a real environment of "working together" builds up during the assignment. While the first task of the first activity requires the student to compile and run the client and server on their local machine the following task to exchange IP numbers and to run the application distributed means that the good students usually start to help their peers in running their application (motivated by their need to find a partner where they can access a server). We see here that this activity is also valuable for enhancing "soft skills" like team working and communication.

In the second activity, running a simple chat program, the students often find it surprising that they are able to implement a service they know from their daily experience (text messaging) from scratch. A small number is often motivated to develop their software further in order to mimic features they know from other software.

7. Conclusion

By selecting out key concepts and fundamental skills (such as RPC's) and by providing just enough industrial context, it is possible to marry vocational content with conceptual development, whilst also motivating students by relating key skills and concepts to the real world they see around them. The material presented here serves as an example on how web services can be introduced in a Java based curriculum providing the students with a positive experience on Web Services.

8. References

- Apache (2001-2003), *About Apache XML-RPC,* http://ws.apache.org/xmlrpc/
- Apache (2002-2004), *The Apache XML Project*, http://xml.apache.org/
- Conrad, Marc (2003), *Lecture slides and activities on XML-RPC,* http://perisic.com/xmlrpc.
- Draganova, Christina (2003), *Web Services and Java,* JICC 7, http://www.ics.ltsn.ac.uk/pub/jicc7/draganova2.doc.
- Sun Microsystems, Inc (2002), *Web Services Made Easier: The Java™ APIs and Architectures for XML, A Technical White Paper*, http://java.sun.com/xml/webservices.pdf
- Sun Microdystems, Inc (2004), *Java 2 Platform, Standard Edition version 1.4.2,* http://java.sun.com/j2se/1.4.2/
- UserLand Software inc (1998-2004), *The XMLRPC homepage*, http://xmlrpc.com..
- W3C (2003), *SOAP Version 1.2 Part 1: Messaging Framework*, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
- Wikipedia, the free Encyclopaedia, *XML-RPC*, http://en.wikipedia.org/wiki/XML-RPC (accessed January 5, 2005)

Teaching Computer Science with the SSCLI

Dr David Grey, Rob Miles
Department of Computer Science,
University of Hull,
Hull, U.K.

Abstract

The Microsoft Shared Source Common Language Infrastructure (SSCLI) is a contemporary managed-code environment which is freely distributed in source code format. It makes an ideal vehicle for teaching various aspects of computer science and software development. This paper discusses how the SSCLI is used to support teaching on a Masters programme at the University of Hull and examines a practical exercise based on it.

1 Introduction

Microsoft's .NET Framework provides a cross-language, component-based, managed code environment. Implemented on top of the Windows operating system it provides a rich, fully featured base class library with classes that enable the creation of a wide variety of software applications, from those with graphical interfaces, through to Web services, Web applications and enterprise-scale distributed applications. The Visual Studio development environment supports the .NET Framework and provides a rich set of tools including compilers for C# and three other languages, debuggers, intermediate code assemblers and disassemblers, security configuration tools, etc.

The technology is underpinned by the Common Language Infrastructure (CLI) [1,6,7], which provides the environment within which .NET programs can execute. The Shared Source Common Language Infrastructure (SSCLI) [1,2] is a shared-source implementation of this component and the associated class libraries. Produced by Microsoft, it is freely available for non-commercial work and is derived from the same code as the commercial .NET Framework.

2 How It All Began

A previous examination of C# and the .NET Framework revealed that it made an excellent vehicle for the practical content of many aspects of a Computer Science degree. As a result of a contact made at a previous JICC conference, the authors were invited to attend the launch event for the SSCLI. During those few days much was learnt about the SSCLI and its implementation. It was immediately apparent that it too was eminently suited to computer science teaching and provided a valuable example of a real, large-scale software development project that could be used to help teach students about the issues involved in developing and maintaining software in the real world.

3 The .NET Framework and the SSCLI

The .NET Framework is a cross-language, component-based, managed code environment implemented on the Windows operating system which supports the creation and deployment of a wide variety of software applications. It is freely available for download and includes a range of development tools, including a debugger and compiler for the C# language. Visual Studio .NET is the latest version of Microsoft's development environment. When used in conjunction with the

.NET Framework it includes additional development tools and managed code compilers for several languages, including Visual Basic, C++ and Java (J#).

4 The Common Language Infrastructure and standardisation

At the base of any .NET implementation is the *Common Language Infrastructure.* In the past many people have accused Microsoft of flouting existing industry standards by producing their own variations on them which, because of Microsoft's dominant market position, become de facto 'standards'. When the .NET Framework was launched Microsoft decided to make large portions of the execution environment, the type system and the C# language the basis of international standards. As a result Microsoft submitted applications to the European Computer Manufacturers Association (ECMA) and International Standards Organisation (ISO).

The ECMA 335 (CLI) / ISO/IEC 23271/2 (CLI) [6] specification defines a language agnostic data-driven virtual execution environment in which 'blobs' of data (components) are brought together as self-assembling, type-safe software systems. It also defines a processor independent, object-oriented Common Intermediate Language (CIL) for describing programs that makes use of a common type system to define its basic types. All programs written in high-level language must be compiled down to a CIL representation before execution. This intermediate language has a rich set of instructions which are used to drive a stack-based abstract machine.

The ECMA 334 (C#)/ ISO/IEC 23270 (C#) [5] standard defines the C# language which is seen by Microsoft as a keystone of the .NET environment. By making these technologies open standards Microsoft has paved the way for other vendors to produce their own implementations for whichever platforms they choose. Implementations which fully comply with the specifications are guaranteed to be completely interoperable with Microsoft's own implementations of these standards.

Much of the remaining portions of the .NET Framework are based on open, existing international standards and protocols e.g. XML, HTML, HTTP, SOAP, etc and industry standard cryptographic algorithms and techniques.

4.1 Differences between the .NET Framework and the SSCLI

Microsoft currently offers three implementations of the ECMA/ISO standards; the .NET Framework aimed at Windows desktop/server platforms, the .NET Compact Framework, aimed at Windows mobile/embedded platforms and the SSCLI. The .NET Framework and .NET Compact Framework are supersets of the ECMA/ISO specifications, the SSCLI is a superset of the ECMA/ISO standards but a subset of the .NET Framework.

The .NET Framework is targeted only at Windows platforms and provides a rich, fully featured base class library which supports the creation of graphical interfaces, Web interactions, Web services, database access and XML manipulation to name but a few. A command line debugger is provided together with a number of useful SDK tools that allow the signing of assemblies, inspection of metadata, assembly and disassembly of intermediate language code, and so on. . Addition compilers and development are provided by Visual Studio .NET.

The Shared Source CLI can be downloaded from Microsoft [2] and contains a complete, freely modifiable and redistributable implementation of the CLI specification in source code form [1]. The SSCLI has been designed for portability and will build and run on Windows 32, Mac OS/x and FreeBSD systems. It has a number of differences from the commercial .NET Framework, particularly with regards to its base class library which is much smaller and omits much of the Windows and Microsoft specific functionality. Specifically is has no support for creating graphical interfaces, database interactions, Web services and Web applications.

The SSCLI was initially derived from an early build of the commercial .NET Framework. Certain features were removed from the source and it was modified to make it more portable. Consequently some of the core components, such as the JIT compiler and garbage collector have been re-implemented to aid portability and to avoid exposing commercially sensitive implementation details. A key goal of the SSCLI initiative is to allow the SSCLI to be ported to other, non-Windows platforms. At present the SSCLI runs on Windows 32, FreeBSD and Mac OS/X platforms. The SSCLI is provided with the same set of SDK tools as the full .NET Framework. C# and JScript compilers are also included in source form.

Microsoft's primary motivation for releasing the SSCLI was as a vehicle for research and to foster learning about the CLI and the technologies required to implement it. A significant number of research projects have used the SSCLI and there are features now available within the SSCLI which are not yet implemented in the commercial .NET Framework e.g. generics.

4.2 Why Use the SSCLI?

There are a number of reasons why one may choose to work with the SSCLI. Firstly it is free and provides all the source code of a CLI implementation which is very similar to that of the .NET framework. It is useful for understanding in detail how the .NET Framework operates; many parts of the SSCLI, particularly the base class libraries, are virtually identical to the commercial framework. If you want to modify or extend the behaviour of a CLI implementation, the SSCLI is currently the most complete implementation available in source format. Finally, if you want to improve your commercial code development skills, the SSCLI is a large software system implemented by Microsoft's developers to the same standards as their other commercial offerings. A lot can be learnt about real-world software development techniques by examining the source of the SSCLI.

5 The SSCLI itself as A Teaching Tool

In the latest release the SSCLI source has over 3 million lines of code and is an excellent example of a large, commercial standard, software product that can teach much about the techniques and skills required for commercial software development. For example, the build process is particularly complex and is typical of the build process of many large systems; it is a classic example of the way software is compiled and built in the real world. An extensive suite of quality test and build-verification tests is included in the distribution and this can be used to expose students to the type and extent of tests that the software industry is required to undertake. Students will undoubtedly learn much that will improve their own software development skills simply by studying the SSCLI code and considering how and why it is put together in the way it is.

6 A .NET-based Masters Programme

The University of Hull had for some time considered introducing an MSc programme which focussed on contemporary distributed systems and their implementation in modern managed code environments such as Java. The opinions of industrialists as employers of graduates from UK computer science degrees had also been sought. They highlighted a number of issues which they felt should be addressed, particularly relating to the lack of exposure that graduates had to large-scale software systems, the development techniques associated with them, experience in software maintenance and a lack of general practical skills in the debugging and testing of software. Developer skills are considered a key component of a well rounded software practitioner. [4]

Having considered these influences it was decided to introduce an MSc course which would expose students to the issues involved in contemporary enterprise-scale distributed system development, the managed code environments used in implementing them and the practical skills

and experience required to create and maintain such systems. .NET and the SSCLI seemed to offer suitable practical platforms for exploring many of the issues which this course would cover so we approached Microsoft and sought their assistance. Through the input of Microsoft UK and the product teams which had developed .NET and the SSCLI, the authors introduced the first degree programme in the world which used .NET/SSCLI as its sole exemplar technologies. The result was the MSc in Distributed Systems Development which we began teaching in September 2003.

The MSc in Distributed Systems Development aims to give advanced coverage of modern distributed computing techniques, enhance skills used for working with large codebases, provide hands-on practical experience underpinned by advanced theoretical concepts and to develop "active practitioners". Throughout the programme, the .NET Framework provides an overarching example of distributed systems concepts and techniques whilst the SSCLI is used to provide "down to the metal" implementation details. In particular, practical exercises in the *Introduction to .NET*, *Maintaining Large Software Systems* and *Virtual Machine Architectures* modules make extensive use of the SSCLI to enhance development skills and to explore the implementation of managed environments. These exercises require students to modify components of the SSCLI such as the JScript compiler and the JIT compiler. We have found the SSCLI to be an excellent environment for enhancing testing and debugging skills and graduates of the programme have commented that their software development skills have improved enormously as a result.

7 Deployment Considerations for the SSCLI

There is no doubt that the SSCLI is an excellent teaching tool. However there are some "interesting" challenges that must be overcome when deploying it in a University teaching the environment.

The practical exercises in which the SSCLI has been deployed all require students to modify and rebuild the SSCLI source. This requires each student to have their own installed copy of the SSCLI and they must build it before attempting any of the exercises. Once installed and fully compiled, the SSCLI require nearly 600MB of hard disk space. Given the size of current hard drives and the numbers of students on an MSc programme, it is not a significant problem to store a copy of the SSCLI in each students account space on a networked server. The SSCLI distribution contains 6000 individual sources files and there can be significant impact on available network bandwidth when a number of students start a build simultaneously. Building on a networked drive takes significantly longer than on a local drive, and a complete local build can take 15 minutes or more.

The biggest challenge faced occurs with course modules that are taught simultaneously, each of which requires the student to work with and modify specific aspects of the SSCLI. In this scenario there is a real danger that work done with the SSCLI in one module could interfere with work done with the SSCLI in another module. For example, one module may require students to modify the JIT compiler; if the student only has a single SSCLI installation in their account, then these modifications will affect every other program run under the SSCLI, including work performed on other course modules. An obvious solution is for the student to maintain a SSCLI installation for each module but at 600MB a time, this does start to have an impact on available storage space and backup mechanisms. An alternative approach which we adopted was to require the students to use a source code control system, such as CVS, to store their SSCLI sources so that they could make changes, restore original files or open any one of a number of different versions. Although there are valid pedagogical reasons for encouraging students to use source control, storing the entire source tree for each student does not solve any of the storage space issues and the administrative overhead in managing the source control system was prohibitive. A strategy of requiring the students to make their own backup of any area of the SSCLI source on

which they are working has since been adopted. Should students need to restore the SSCLI to its original (or any other) state they can do so from their backups. A clean copy of the SSCLI distribution is also made available from which they can restore a vanilla installation. Although not ideal, this approach minimises administrative overhead and storage space concerns and so far no major problems have been experienced.

8 A Sample Exercise: Enhancing Debugging Skills

The ability to test and debug programmes is an essential skill for computer scientists and software developers. The presentation which accompanies this paper gives a live walkthrough of one of the exercises from the *Maintaining Large Software Systems* module of the MSc in Distributed Systems Development. This exercise is designed to help students develop debugging and critical reasoning skills, and to challenge some of their assumptions about the tools and technologies they use. The complete exercise is available in the public domain [3] and can be freely used in other computing courses.

In the exercise, students are asked to evaluate a bubblesort routine in C# which must sort an array of supplied floating point values. This program is compiled against, and executed on, the SSCLI and will not work correctly with the supplied test data due to a deliberately introduced fault in the SSCLI C# compiler.

Teaching what to do when it all goes wrong is difficult. Throughout a degree programme students will have to debug their own programs but will spend little time debugging code written by others, whereas in industry much of their work will involve debugging and maintaining code written by others. In this exercise students must debug code they have not written and are faced with a "worst case" scenario. There is a problem with the underlying compiler implementation not with their bubblesort program; the program is fine, but it still doesn't work!

The students need to decide if the source program is at fault. The symptoms of the introduced fault are similar to those encountered by inexperienced programmers relating to problems with array bounds. The students must first convince themselves that the program is correct and they will not expect the language implementation to cause problems.

The exercise is carefully structured and leads the students through a systematic process of constructing and testing hypotheses. It begins by getting the students to prove the correctness of the code by running it under the commercial .NET Framework, where it works correctly. From this they should deduce that an issue with the SSCLI environment is causing the problem. They are then led through the construction and testing of further hypotheses before arriving at the conclusion that there is something wrong with the < operator. This leads to the hypotheses that perhaps the compiler is generating incorrect intermediate language (IL) code from the C# source. The students are shown how to investigate this by using the IL disassembler to inspect the IL produced by the compiler and using the IL assembler to hand modify the IL code in an attempt to fix the problem. Once they arrive at the conclusion that the compiler creates an off-by-one error in the code it outputs for the floating point < operator, they are then lead them through the steps needed to debug the source of the C# compiler and rectify the problem.

For most students this is a challenging exercise and it confronts some common preconceptions. Most new graduates would not produce sufficient test cases to narrow the problem down to the floating point comparison, nor would they suspect the compiler of producing incorrect code. The SSCLI source code is large and unfamiliar to them and this causes difficulties for many of them. Yet this is exactly the situation that they will face when undertaking software development and maintenance for real.

The exercise presented here is just one exercise from this module. As a result of this and similar exercises, students develop an appreciation of the importance of a methodical approach and gain skills in the design of test cases, navigating unfamiliar code bases and using debugging and other development tools in an effective manner.

9 Conclusions

Having spent three years working with .NET and the SSCLI, the authors are firmly convinced of their value as tools for teaching various aspects of computer science and software development. The SSCLI is particularly valuable as an example of how software development is done in the real world.

The MSc in Distributed Systems Development is now in its second year; it has attracted much interest from around the world and has received favourable comments from the students who have studied it. The SSCLI and .NET practical exercises set on this programme have proved challenging to the students but they have all enjoyed these exercises and found them useful in enhancing their software skills. The authors have also found teaching this course to be both a challenging and thoroughly enjoyable experience.

The involvement of Microsoft, and other industrialists, was critically important to this development of this course. This gave access to Microsoft engineers and technical materials which helped in the development of some of the more detailed technical aspects of the course and helped to ensure that the content of the course is directly relevant to the needs of industry. The authors believe that this degree significantly enhances the abilities of students and builds on what they have learnt in an undergraduate course to make them into more fully rounded software practitioners.

References

[1] *The Microsoft Shared Source CLI Implementation*, Microsoft Corporation, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/Dndotnet/html/mssharsourcecli.asp

[2] *The Microsoft Shared Source CLI*, Microsoft Corporation, http://msdn.microsoft.com/net/sscli/

[3] *Teaching Software Maintenance using .NET and Rotor*, University of Hull, http://www2.dcs.hull.ac.uk/people/cssdjg/Courses/Teaching%20Software%20Maintenance%20Using%20.NET%20and%20Rotor/index.htm

[4] *Code Complete*, 2nd Edition, Steve McConnell, Microsoft Press, 2004

[5] *Standard ECMA-334 - C# Language Specification* 2nd edition December 2002, http://www.ecma-international.org/publications/standards/Ecma-334.htm

[6] *Standard ECMA-335 -Common Language Infrastructure (CLI)* 2nd edition December 2002, http://www.ecma-international.org/publications/standards/Ecma-335.htm

[7] *The Common Language Infrastructure Annotated Standard*, James S Miller & Susann Ragdale, Addison-Wesley, 2004

[8] *Essential CVS*, Jennifer Vesperman, O'Reilly, 2003

INTERNET LED PROJECT PREPARATION – USING PROBLEM BASED E-LEARNING IN THE CLASSROOM

Thomas Lancaster & Cain Evans,
School of Computing and Information,
University of Central England,
Perry Barr,
Birmingham B42 2SU,
UK.
thomas.lancaster@uce.ac.uk & cain.evans@uce.ac.uk

ABSTRACT

The paper describes an e-learning module for second year computing students aimed at preparing them to undertake an advanced final year project. The module takes an alternative strategy for project preparation employing problem based e-learning approaches to guide students to find out how to approach a project relevant to the particular area of computing that their route is focused on. The results are presented as a case study about using the Moodle virtual learning environment (MVLE) to manage large classes where students have multiple individual goals. A number of relevant issues for successful implementation of an e-learning module that have been identified during the delivery are also given.

1. INTRODUCTION

A common feature of UK Computing courses accredited by the British Computer Society is that final year students have to complete an advanced project. Such a project has to involve the student with finding a solution to a practical problem for a designated customer, submitting this as a deliverable and additionally compiling in a written project report detailing the process followed and the developmental considerations made. Projects can be expected to form a substantial contribution towards a student's final degree classification and hence it is necessary to ensure that they are well prepared.

One problem inherent with project preparation modules is that they need to prepare students on a wide variety of degree routes to complete projects suitable for their route, usually with limited resources available. The believed most common method used prepare students with a range of generic skills, such as how to produce a time plan, research literature or fill in a project proposal. It can be argued the appropriateness of this preparation is very limited as students will not have had the experience to carry out a project relevant to the route they are studying. Students studying a Software Engineering route, relying on an implementation, could be expected to require a vastly different set of skills to students on an Information Studies route, more related to human and organisational issues.

This paper describes and provides an initial evaluation of an alternative problem based e-learning approach, so called because students are largely discovering for themselves how to complete a structured project, with the time and organisational issues that go with it in order to give them hands on experience. Problem based e-learning is becoming an increasingly used pedagogical design (Mettleweb 2004). The implementation used sees students introduced to the ideas of discovering information for themselves early, by focused group discussions, assessed using a combination of patchwork and peer review. The major component of this experimental project preparation approach (EPPA) is for students to complete a project in the

same style as one undertaken in the final year albeit in a group rather than an individually. It is hoped that this case study of the approach followed will prove useful to tutors looking to evaluate alternative methods of project preparation or to develop other modules using innovative e-learning strategies. It is also intended that the availability of the EPPAs will inspire further development in a project preparation process that may have become largely stagnant.

2. AVAILABILITY OF VLEs

Delivering a Web-based module usually requires the use of an appropriate virtual learning environment (VLE). There has been a recent surge in the number of e-Learning software packages both commercial and open-source and many institutions are now able to offer one or more of these for their tutors. The discussion of which VLE to use can usually be assumed to be made at the institutional level; that is individual tutors cannot decide which would be most appropriate for their own scenarios.

Table 1 lists the major VLEs and e-Learning software applications that have been made available either through open-source licences or commercially, including a url from where interested parties can find out more details of each of these.

Product	Organisation	URL
Learning space	Lotus Education of Lotus Institute	http://www.lotus.com/
WebCT	WebCT, Univ. British Columbia	http://www.webct.com/
TopClass	WBT Systems	http://www.wbtsystems.com/
Virtual -U	Virtual Learning Environments Inc.	http://www.vlei.com/
Web Course in a Box	MadDuck Technologies	http://www.madduck.com/
Asymetrix Librarian	Asymetrix	http://www.asymetrix.com/
FirstClass Classrooms	SoftArc	http://www.softarc.com/
CourseInfo	Blackboard Inc	http://www.softarc.com/
ARIADNE	EPF Lausanne (EC DG XIII)	http://ariadne.unil.ch/tools/
CoMentor	Huddersfield University	http://comentor.hud.ac.uk
CoSE	Staffordshire University	http://www.staffs.ac.uk/cose
Learning Landscapes	TOOMOL Project, UW - Bangor	http://toomol.bangor.ac.uk
ATutor	ATRC Corp.	http://www.atutor.ca
Learning space	Lotus Education of Lotus Institute	http://www.lotus.com/
WebCT	WebCT, Univ. British Columbia	http://www.webct.com/

Table 1 - Virtual Learning Environments: adapted from JISC Technology Applications Programme (JISC 2005).

It is neither possible, nor desirable, to describe all of these within the confines of this paper. Instead this background will focus on an illustrative three VLEs.

WebCT (2004) is a commercial provider of e-learning systems for higher education institutions. WebCT is said to provide a flexible e-learning environment that enables institutions the ability to create and distribute innovative teaching environments. It is claimed that thousands of institutions in more than 70 countries are using WebCT's e-learning solutions to expand the boundaries of teaching and learning..

Moodle (2004) is a freely available open source course management system (CMS), designed specifically to manage Internet-based educational courses. Environments for where is said to be suitable range from fairly restrictive Web hosting providers to institutional servers. It

contains complete support for MySQL and PostgreSQL, and support for other databases is very easy to add. Only a single database is required, and Moodle can share this with other applications. A modular authentication design allows Moodle to be connected to external authentication services such as LDAP, IMAP, POP3, and NNTP. Moodle also supports traditional email authentication and instant guest accounts. Moodle's other key feature is said to be customisation. This wide-ranging concept encompasses ideas from the formatting of the look of an individual module, to naming conventions (e.g. instead of 'teacher' and 'student' a tutor might prefer using 'professor' and 'participant').

ATutor (2005) is an alternative Open Source Web-based Learning Content Management System (LCMS). Key strengths of ATutor are said to include administration, accessibility and adaptability. Educators are said to be able to quickly assemble, package, and redistribute Web-based instructional content, easily retrieve and import pre-packaged content, and conduct their courses online. Students are said to learn in an adaptive learning environment.

The authors' institution has adopted the use of Moodle. As such any in-depth attempts to compare Moodle with the other VLEs will be inherently limited. Some may argue that the choice of Moodle is because the institution is a keen proponent of the open source software movement; others may give a more cynical view relating to the costs of open source software. From the point of view of the authors, the ease with which the implementation of Moodle can be altered to suit a different purpose is a key benefit of this VLE. The remainder of the paper will look at the delivery of a particular module in Moodle, allowing potential users to make a decision about its appropriateness for their own needs.

3. ADMINISTRATION OF A MOODLE PROJECT PREPARATION MODULE

The A2C module (IT Professional Practice 1: Project Preparation) is delivered three times during any academic year, with students getting a choice of studying A2C in the first or second semester and given a retake opportunity during the summer. A2C is currently mid-way through its initial run. A generic framework is in place to allow A2C to be delivered electronically in a standard format so that students cannot claim to be disadvantaged based on when they chose to study it. Modified assessment tasks on a theme relevant to the particular students studying A2C can be blended into this delivery framework. Students are guided towards successful completion of these by using the electronic resources available.

The Moodle virtual learning environment (MVLE) is being used to deliver the A2C EPPA module. Although Moodle has an active team of developers and a support community tutors are not required to have any technical knowledge in order to use Moodle to deliver a module. Examples of MVLEs are available from around the world (Yeovil 2004, UCE 2004). Figure 1 shows Moodle in use to deliver A2C and demonstrates some of the features the MVLE makes available.

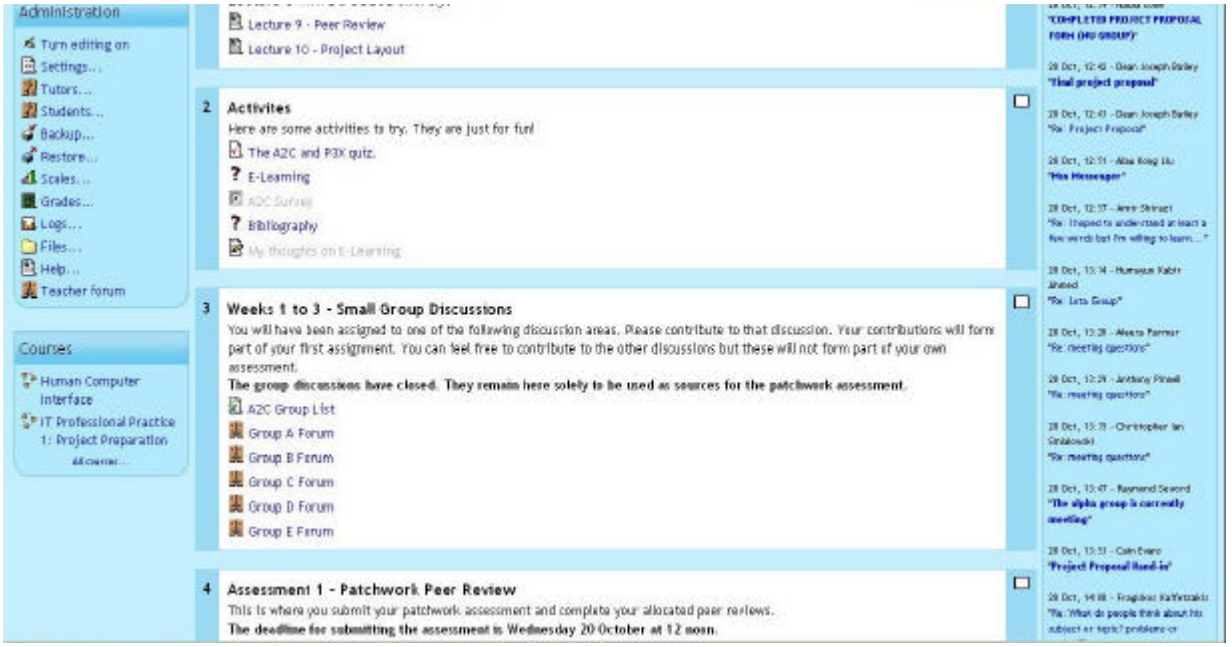


Figure 1 – Moodle in use

Excluding materials intended to aid students with assessments, participants on A2C have been provided with a number of common resources to enable them to get the most out of the module. There are currently ten video lectures, delivered using Microsoft Producer and Pinnacle's Liquid package, with the intention to add more as and when students appear to need further knowledge. These lectures a combination of 'talking heads' (Baskins 2004) with integrated Powerpoint slides and each last around ten minutes. Lectures cover such topics as completing a project proposal, or research methods. Full module and assignment specifications are provided, with Web links, where appropriate, to allow students to find out more information. A forum is provided for questions of interest to the whole group, with those likely to be of continued interest collected into a Frequently Asked Questions (FAQs) file for future delivery of A2C. A number of optional formative activities such as quizzes are implemented to allow students to test their knowledge of a given subject. Finally students are provided with an off-topic forum and chatroom to allow a community to be developed. So far there has been no abuse of these resources.

The administrative facilities available to tutors are also of interest. At any time tutors can view which students are online and check which activities that they have engaged in the form of logs and charts. Tutors have their own private forum to allow for the co-ordination of multiple classes that are studying similar materials. There are simple ways of tracking student marks. One final feature of note is the Moodle implementation of groups, where students can be split into sub-classes to work on material as if in an independent seminar classroom setting. This means that they should not see the contributions of other sub-groups. The exact implementation of this has meant that it hasn't been useful for the delivery A2C, instead students have been allowed to view all contributions, with carefully worded instructions given to ensure that they only post to their designated forums.

4. PATCHWORK PEER REVIEW ASSESSMENT

A2C is structured around two assessments, both managed using Moodle and running sequentially one after another, albeit with a few days of unavoidable overlap for a process of peer review. The module is entirely assessed by coursework. The implementation of the EPPA ensures that students are working on assessed tasks throughout their time on A2C.

The first assessment covers weeks one to four of the module, with a small amount of overlap with assessment two during week five. Initially students are allocated to a discussion group, of around twenty students; during the first run of A2C five groups were necessary. Each group is issued with a tutor to take the role of facilitator to guide them through a problem based learning approach. The topics are chosen with suitable depth to allow academic research to be identified and relevant questions discovered. Examples of topics issued during the first run of A2C include automated personalisation of news Web sites and spam prevention techniques. All groups are led on the same structured discussion, identifying issues, finding and evaluating a relevant research paper and coming up with a possible research question.

The associated assessment takes the form of a patchwork (Patchwork Text 2003), to be assessed using peer review. Here a patchwork is defined as a combination of comments made by students in discussion forums, known as 'patches', combined with original and appropriate linking material, known as 'stitches'. Students are told they can use patches of either their own forum contributions, or those of other students, as appropriate. They are advised to use a mixture of both, properly referenced, to make up around 50% of their submission. Students are expected to summarise their forum topic and find a research question from a relevant paper. For students who had participated in the forum process as expected the majority of this work should already be complete. This implementation of patchwork is intended to allow students who have not contributed fully, or who see more mileage in a question identified by another student, to reference and use their question. The key additional part of the patchwork assessment asks students to suggest a solution to the question that they chose, or give ways of finding a solution or evaluating a solution. Students should see that this mirrors the process of preparing to complete a project.

The peer review process then forms the final component of this assessment, using the MVLE's Workshop module (Moodle 2004). Each student is asked to review the work of five of their peers, comprising both their contributions to designated forums and the quality of their patchwork. A marking scheme is made available to students at all stages of the process. They are also asked to assess their own contribution. The mark awarded to each student for assessment one comprises 75% based on the reviews they receive from their peers and 25% based on the accuracy of the reviews the student supplies, based around the mean mark supplied for each student.

The general impression of the A2C tutors during the first run of this module is that this process was successful. Students identified a range of relevant questions within their specific areas and suggested some good solutions. They also appreciated being able to look at the work submitted by other students, something that they rarely, if ever, had a chance to do in other modules.

The main concern expressed by students was about the reliability of the peer review process and whether the lack of anonymity would mean that they would be disadvantaged. The weighting of the two components of the assessment, in particular giving 25% for the accuracy of the peer review, was chosen deliberately to avoid this, as anyone awarding intentionally high marks to friends, or low marks to students who they did not like would receive a low accuracy component of their mark. This approach seems to have worked successfully. Generally the marks awarded for assessment one have been observed to be higher than the student's received in other modules and this reflects their greater levels of involvement in the learning process.

5. GROUP PROJECT

The remaining 75% of the marks for A2C are awarded for students completing a project in the style of a final year project in groups of around six, an advised method of e-learning assessment (Adelaide 2004). In common with final year projects project skeletons are allocated to groups, rather than allowing students to come up with their own, however the ideas given are fairly open ended to allow students to develop them in the way that they feel most appropriate. The allocation of groups and corresponding projects is based around each individual student's choice of route, where the students make this available. For the current run this has meant the provision of around 16 problem solving project ideas in skeletal form, each with a designated customer and supervisor. These people are contactable primarily through a supplied Moodle forum, in future this may allow alternative personas to be identified for academic staff to take on the role of customers over the MVLE.

Completing the equivalent of a final year project in a much-limited time, even as a group rather than an individual is an involved task and so students are warned of this. As such each group needs a level of planning and sensible distribution of work. The projects issued are chosen to reflect the more limited time available but to allow experience of all stages of the project process. The marking scheme and number of forms to be completed is also relaxed from that used in the final year. During the current run examples of projects issued include generating text in the style of a chosen writer as a Software Engineering project and implementing an appropriate touch-typing instructional CD for a Multimedia project.

In order to facilitate the completion of a project in the time available and to ensure that the final year project process is reflected as far as possible groups are issued two formal appointments with their supervisor. The first of these, worth 10% of the assignment two mark, is a discussion of the group's project proposal. Groups are expected to have used the discussion area provided to identify any additional customer requirements not stated in the skeleton outline and to ensure that a realistic time plan has been produced. The second, a final project viva worth 20%, is there to allow the supervisor to check that the work submitted is the group's own and to provide a debriefing session for the students in advance of their final year project. It is anticipated that some projects will go wrong and this is not necessarily a criteria for failure of the module, merely to make students more prepared to avoid something similar for their final year.

The remainder of the marks for assignment two are allocated as follows. 50% of the marks are for the quality of the main report and the deliverables produced, marked using the same scheme used in the final year. The other 20% are based around student's evaluation of the other members of their group. This is intended to allow groups who have progressed well to receive a high overall mark, whilst ensuring that any group members who have not contributed fully do not benefit to the same extent as their peers.

A full evaluation of the second assessment is not yet available, as it is currently underway. The initial indications are that it has been well received and is progressing well, however there are a small number of students who are not engaging with the rest of their group. The inclusion of an anonymous vote that allows a majority of participating students to give zero to any student for a given component of the assessment has allowed most of these problem students to be identified.

6. ISSUES FOR SUCCESSFUL E-LEARNING IMPLEMENTATION

The experience of the tutors involved with delivering A2C has allowed them to identify a number of issues that they faced that may affect the successful delivery of an e-learning module. It is apparent that the number of different educational technologies available for higher education tutors has increased rapidly during recent years. More tools and development environments seem to be being made available to teachers. Since the Dearing Report (Dearing 1997) higher education institutions have looked to adopt ICT into the classroom for instructional usage. It is suggested that combining ICT with traditional teaching is not a task to be implemented lightly. There is a need for infrastructure and management support to continue the efforts of combining online and offline learning. There are a number of key issues that directly affect the success of an e-Learning system such as the one developed at UCE for the A2C module.

Issues that could affect the successful implementation or, indeed, its failure may depend on:

1. Resources made available at the immediate environment, e.g., computer hardware/software access;

2. Infrastructure requires updating in relation to building servers for the location of the e-learning software;

3. A concrete policy for online learning (e-Learning) is needed at a higher level than the Department or School, i.e., a unilateral institutional policy that states clear objectives for using e-Learning and Virtual Learning Environments (VLE);

4. Collaboration between Faculties/Departments is also a potential knowledge sharing opportunity to learn from each others' experiences; and

5. The provision of raining and knowledge sharing events as a useful and meaningful way to reflect on problems and solutions.

Given the complexity of transforming traditional teaching it is envisaged that there will be teething problems, as were the case with the developments of A2C. However it is the view of the authors that, as and when these initial problems are dealt with, the returns of using VLEs are vastly greater than those from sticking to traditional pedagogical methods. Nevertheless, the learning curve could be considered fairly steep for those with little or no IT background. With appropriate training this shouldn't deter anyone from combining and implanting new and innovative ICT methods to overhaul teaching.

7. EVALUATION OF MODULE DELIVERY METHOD

The first run of the A2C module and indeed of the delivery of a new style of projects that have not yet been taken by final year students is currently in progress. As such this paper has been presented primarily as a case study and as a proposed methodology, which, it is hoped, will allow tutors to implement problem based e-learning modules successfully. Formal and informal feedback has been encouraged from students regularly and where appropriate this has been discussed under sections 4 and 5 of this paper. This final section will discuss the success of A2C from the tutor perspective and any changes that are under consideration in order to make the EPPA more suited for the student learning experience.

An initial observation is that, from the perspective of A2C tutors, e-learning modules have to be developed as such from the outset. Successful delivery of A2C required a revalidation of its learning outcomes to ensure that they were in line with what could be delivered using the MVLE and what was needed to prepare a wide variety of students with different learning styles for different types of projects. It is vital to not have unrealistic expectations about what the e-learning environment can be used for. In the case of A2C the tutors found that, although the groups feature did not work as anticipated, suitable amendments were possible. Generally the MVLE has been found to be reliable, intuitive to use and has the big benefit of being open source, freeing institutions from licensing costs and allowing them to develop further features if these are deemed necessary.

The volume of work required to develop and administrate a successful e-learning module should not be understated. The initial run of A2C has required a large amount of tutor resources to develop materials such as video lectures, suitable assignment tasks and to answer unanticipated student queries, not to mention the day-to-day running of the modules. One important design choice made by the tutors was to develop the module within a standard framework; this means that the components of the module and its outline assignment tasks are reusable. All that it is necessary to do to rerun the module is to add appropriate new discussion topics and project ideas within the same overall package, something that should be more manageable for a module that runs thrice yearly.

One of the main administrative issues that has to be dealt with on a piecemeal basis is that of late registering students and trying to fit them into established groups. A proposal has been put forward to run A2C on a basis of continual delivery, so that groups can be allocated as and when required, which should be one of the key advantages of e-learning. There can also be a mismatch between when tutors are available online and when students perceive they should be available. Since there is nothing to stop students working at night and for many students this should be encouraged, students can have unrealistic expectations. It is important for tutors to make known their likely reply pattern early to avoid this.

The decision of using two assessments appears to have been well received, in particular the use of patchwork and peer assessment, two ideas that the students have not met previously. It seems unfortunate that since the first assessment was done so well that this cannot be weighted greater than 25%. The exact weighting of the two assignments is under consideration. In order to facilitate this, to allow students to start their group project faster and to manage rolling starts it is being considered that the ideas for group projects could be based on the initial discussions, splitting each group of around twenty students into three relevant smaller groups. This would involve choosing appropriate initial areas for discussion with scope for projects under every possible final year route, which should be feasible.

Early feedback from students has been good, with students stating that they feel more involved with the module than many others, despite the fact that it is being delivered primarily electronically. The main question still to be answered will be whether the module meets its overall purpose, of preparing students more successfully to complete a final year project. That evaluation will not be possible for another year until the first run of the new final year projects has been completed. If the current success rate of A2C compared to its predecessor is an indicator than all initial signs of this are highly promising.

REFERENCES

Adelaide (2004). Available at http://www.adelaide.edu.au/clpd/staff/online/assessment/methods.html. Accessed 1 November 2004

ATutor (2005). Available at http://www.atutor.ca. Accessed on 5 January 2005.

Dearing (1997), The Dearing Report. Available at http://www.bbc.co.uk/politics97/news/07/0723/dearbrief.shtml. Accessed 5 January 2005.

JISC (2005). Available at http://www.jisc.ac.uk/index.cfm?name=programme_jtap, accessed 5 January 2005.

Mettleweb (2004). Available at http://mettleweb.unimelb.edu.au/guide/pedagogy.html. Accessed 1 November 2004.

Moodle (2004). Available at http://moodle.org. Accessed 1 November 2004.

Patchwork Text (2003). Available at http://www.apu.ac.uk/conferences/patchwork.shtml. Accessed 1 November 2004.

Talking Heads (2004). Available at http://www.haskins.yale.edu/haskins/HEADS/face.html. Accessed 1 November 2004.

TESL-EJ (2004). Available at http://writing.berkeley.edu/TESL-EJ/ej30/m2.html. Accessed 1 November 2004.

UCE (2004). Available at http://artemis.sci.uce.ac.uk/moodle. Accessed 1 November 2004.

Yeovil (2004). Available at http://moodle.yeovil.ac.uk. Accessed 1 November 2004

Observations on the use of a Simple Interactive Development Environment for C# in teaching

Keith Mannock, Trevor Fenner, George Loizou and Marie-Helene Ng Cheong Vee
School of Computer Science and Information Systems,
Birkbeck, University of London
London, WC1E 7HX

Abstract

We discuss the results of a two year study into the use of a lightweight, easy-to-use teaching environment for the C# language. The experimental groups cover undergraduate and taught postgraduate students taking an object oriented programming module. The results suggest that, although such a tool may have certain advantages from an educational perspective, the availability of a premier commercial tool (Visual Studio.NET) outweighed the advantages in the eyes of the students, involved in this study. The study also suggests reluctance on the part of the students to embrace the C# language.

1. INTRODUCTION

In [Fen02] we described a prototype design and implementation for C#ide, a simple interactive development environment (IDE) for C# and .NET. In this paper we discuss our findings based on using this tool over the last two years to aid the teaching of object oriented programming on our undergraduate and master level courses.

2. C#ide (pronounced *seaside*)

One of the fundamental requirements of this work was that we did not want to have to devote a significant amount of time to teaching about the programming environment itself. We saw the need for a lightweight, easy-to-use teaching environment for the C# language, much like BlueJ provides for Java [Kö03]. The Visual Studio .NET IDE is the de facto standard for application development on the Microsoft Windows platform. The main problem we foresaw in using this for teaching[1] was that VS.NET is a complex and daunting prospect for students.

Although not our original intention, the environment we developed for C# was quite similar to the BlueJ environment for Java. The system is an integrated environment with the following characteristics:

- a simple project manager
- a language sensitive editor with syntax colour-coding functionality
- graphical display of the class structure

[1] Licensing costs are now not a major consideration.

- compilation of the code without any additional setup required from the user (i.e., assemblies, projects, etc.)
- easy access to language reference and help facilities
- access to the debugger, which provides simple single step tracing and inspection facilities
- an experimental object workbench with object creation, interrogation and method invocation facilities.

Most of this functionality is provided in the latest versions of BlueJ, so we had effectively created a similar system for C# and .NET (see Figure 1). The original implementations of C#ide were based upon the ROTOR source code [St02], because we intended to provide portability across operating systems. This proved to be an unwanted feature for the students as they chose to use Microsoft Windows for their C# work; for convenience the last version of C#ide was therefore a single platform Microsoft Windows based tool.

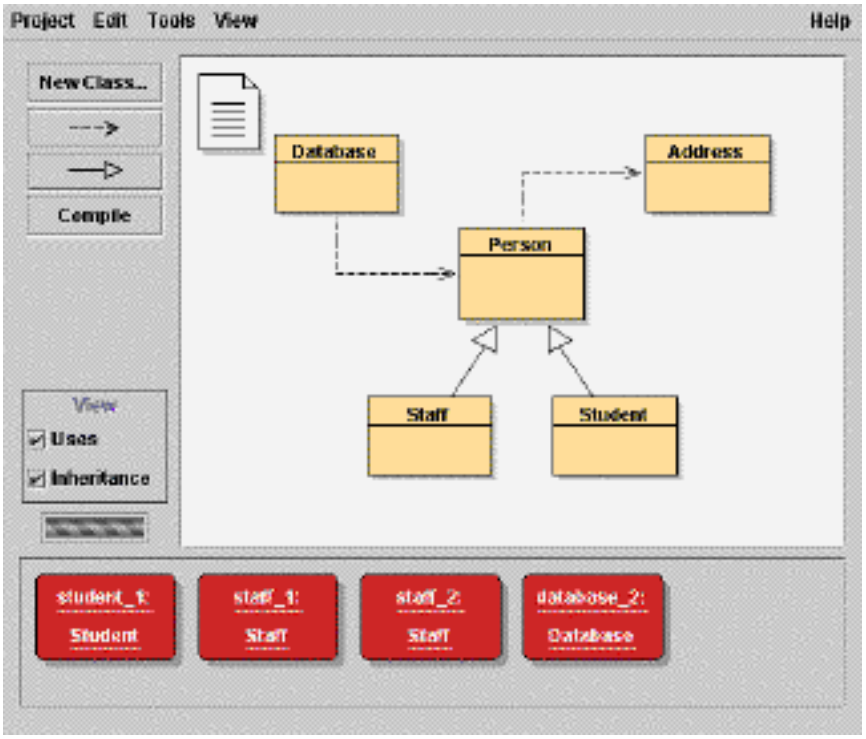


Figure 1

3. THE STUDY

The experimental study was designed to encompass both undergraduate and postgraduate courses over a period of two years. We had identified second courses in programming for the use of C#ide. We would have liked to use the tool in an "objects-first" approach, but the existing structures of the various courses meant that this was not possible. In all cases addressed by our study, we were introducing object oriented constructs in C# using C#ide to an audience that was already familiar with procedural programming in either C++ or Java.

For the undergraduate classes we had intended to introduce C# as part of their second programming module. By that time they should have been comfortable with the basic syntax

of Java and the use of BlueJ. We thought therefore that the transition to C#ide would be fairly painless, given the similarity of the two tools. What we had not expected was that the students still had only a basic knowledge of programming (and Java in particular)[2]. We therefore decided not to risk further confusion and dropped the idea of using C#. In the second year of the study we decided to provide C#ide for use in student projects. This had the advantage that the students would be learning C# themselves as part of their project and would therefore be highly motivated. Also, moving to C# would be facilitated by the similarity of C#ide to BlueJ, with which they were already familiar.

For the postgraduate students we introduced C# as part of the object oriented programming module. This follows on from an introductory programming module that uses C++ but does not describe in detail the object oriented aspects of the language. We had the advantage of two intake groups for the same course within an academic year as the course runs in part-time and full-time modes. This enabled us to run the study in the autumn term with the part-time students and then make appropriate changes before introducing the material to the full-time students in the spring term.

The presentation of C#ide was as part of an "objects" view of the language, as the students were already familiar with procedural constructs from C++. This meant we only needed to describe the mapping of syntax and constructs before introducing the semantic differences between the languages. We used a single case study (a card playing program) to unify the modelling aspects of the course with the object oriented concepts of our module. Then, as a requirement of our syllabus, we introduced Java using BlueJ for the last few weeks of the module. (Given the similarity between the languages we did not consider this too much of a burden for the students.)

4. ANALYSIS OF THE STUDY

There were a number of problems with the approaches we tried. As the majority of our students study part-time and work in the "real world", we thought they would be pleased to be using C#, a cutting edge Microsoft product, especially as they are accustomed to using Microsoft products on a daily basis. The reality of the situation was somewhat different. For reasons of clarity we will discuss the three groups of students, undergraduates doing projects, part-time taught postgraduates and full-time taught postgraduates, separately.

4.1 Part-time taught postgraduate students

The end-of-module questionnaires [Man04] showed that the students, even those with a strong Microsoft bias, preferred to learn a non-proprietary language and environment; it appeared they were happier working with Java. They questioned the choice of C# at all, and said that the contrast between Java and C# was not sufficient to warrant teaching both languages. Out of 53 part-time students surveyed (at the end of the module), all bar two stated that they would have preferred Java as the core language of the module, even if this meant dispensing with C# altogether.

On the issue of the IDE, we had a response that we were not expecting. Although, in the preceding module, their C++ work had been carried out using a command line approach, almost all the students preferred to use VS.NET, despite its complexity. Those who did not use VS.NET continued to use command line compilation and execution in preference to C#ide. We did not promote any particular environment, but made the case for each of the different approaches (including SharpDevelop [Ho03]).

[2] We should mention at this point that the students are not majoring in Computer Science.

4.2 Full-time taught postgraduate students

For this group of students we could benefit from our experience with the part-time students. We therefore decided to start with Java and then offer the students an early switch to C#. It transpired that the students wanted to know more about Java and advanced object oriented programming issues. This meant that the C# component of the module was relegated to an optional set of lectures. Those students who attended the optional lectures (actually most of the class) preferred to use VS.NET as they saw this (but not C#ide) as a useful addition to their CVs.

In defence of C#ide, this group of students, in the main, preferred not to use BlueJ for Java; it was therefore not too surprising that they did not want to use C#ide for C#. The students used either text editors and the command line or Eclipse [Ho04] for the compilation and execution of their Java programs. Their comments in the end-of-module survey indicated that 5 out of 28 students used BlueJ for Java; no one admitted to using C#ide for C#.

4.3 Part-time undergraduate project students

We had hoped, given the self motivational aspect of having to learn C#, that this would be a good cohort for the use of C#ide. We were somewhat surprised by the results. Remembering that these are not Computer Science majors, we thought that the visual nature of C#ide would be an attractive proposition for working with a new language. This view was reinforced by the fact that their only experience of programming on the degree course was of using Java with BlueJ.

None of the project students wanted to use C#ide. They decided that, as they were learning a Microsoft language, they should also use the de facto standard environment for developing such programs, namely VS.NET. When questioned about this decision after the event, the students said that the advantages of using *wizards* in VS.NET had compensated for the time it took them to learn the environment.

5. CONCLUSIONS AND FURTHER WORK

The rationale for the development of C#ide is detailed in [Fen02], but two of the considerations mentioned there came to be re-examined:

1. Over many years of teaching object-oriented programming, we had become aware of the visual metaphors that students found most helpful for learning object-oriented concepts.
2. Students had difficulty thinking in terms of classes and objects, although structuring the problem via these concepts is one of the most important aspects of object-oriented programming and design.

The students were not bothered about the visualisation of the class structure of their programs. They actually preferred a list of classes that they could expand or collapse as necessary, and this functionality was readily available in VS.NET. They preferred to see the code and wanted the *Intellisense*[3] features of the more complex environment. They also wanted experience of this environment as it was one they would, most probably, be using commercially.

Using C#ide for teaching object oriented programming seemed to have no effect on the students' ability to comprehend classes and objects. They could just as well draw the structures we were dealing with. The surveys, quizzes and coursework supported this view, as using this IDE (or any IDE) did not seem to improve their understanding.

[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsintro7/html/vcovrAutomaticStatementCompletion.asp

The students' experience with BlueJ and Java had led them to believe what we had considered as one of the possible drawbacks when we originally proposed C#ide, viz. that the experience of using an IDE with reduced functionality does not necessarily prepare them for the subsequent move to a more feature-rich IDE. For more proficient programmers, moving directly to the advanced IDE may be no more difficult and more cost effective.

Some of these comments are a consequence of not having used an objects-first approach in the first programming module. We believe we would have had a more successful experience with C#ide on initial programming courses. We also experienced strong resistance to C# in general, even given its commercial roots.

In our original paper [Fen02], we had proposed that the transition from Java to C# would be easier if similar development environments were used. We hoped that the "compare and contrast" approach to the languages would become much simpler, and that the students' comprehension of the language features would be improved. The reality of the study showed that the transition was hardly affected by the IDE at all; the students were quite capable of doing the mapping without any intervention via a tool.

Following our study and the results of the various surveys, the requirement for C#ide came into question and therefore C#ide has been *retired*. Given the above analysis and the advent of cut-down versions of VS.NET, e.g. Visual Studio Express, there seems to be less of a requirement for such a tool. We are still left wondering whether using C#ide might prove more successful for a first programming module, but we do not at present foresee any such change in the syllabi of any of our initial programming modules.

The undergraduate course still uses Java with BlueJ. The postgraduate module has been changed to introduce Java with BlueJ, and then Eiffel; the contrast between these languages being more significant from an object oriented perspective. However, it still remains an open question as to whether a multi-language simplified IDE would be a useful teaching tool.

ACKNOWLEDGEMENTS

This pilot project was supported (in part) by a grant from *Microsoft Research*.

REFERENCES

[Fen02] Fenner, T., Loizou, G., Mannock, K., and Ng Cheong Vee, M-H., *A simple Interactive Development Environment for C#* , JICC7, Jan 2003.

[Ho04] Holzner, S., and Rosenblatt, B., *Eclipse: A Java Developer's Guide*, O'Reilly UK, 2004.

[Kö03] Kölling, M., Quig, B., Patterson, A., and Rosenberg, J., *The BlueJ system and its pedagogy*, Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, Vol 13, No 4, Dec 2003.

[Ho03] Holm, C., Spuida, B., and Kruger, M., *Dissecting a C# Application: Inside SharpDevelop*, Wrox Press Ltd, 2003.

[St02] Stutz, D., *The Microsoft Shared Source CLI Implementation*, Microsoft Corporation, March 2002.

Collaborative Open Learning Environment

Colin B. Price
Computing Section, Business School, University College Worcester,
Worcester, WR2 6AJ, UK. c.price@worc.ac.uk

Abstract **A Java-based Environment for learning and teaching programming is described. This can be deployed on the Internet, facilitating collaborative learning between students situated at different locations. The environment uses autonomous simulated robots as the objects of programming and can also be deployed locally, both on a stand-alone simulator and on cheap physical robots. Identical code runs on the physical and simulated robots. This paper details the local and distributed implementations, the software structure and the pedagogy of the environment.**

1. INTRODUCTION

As CS students progress through their courses they are often required to work with several programming languages and paradigms. Modules covering e.g. Operating Systems, Artificial Intelligence, e-Commerce, embedded systems or simply "Programming" each may require some competence in a different language. Some students find this prospect daunting. The burden can be lightened by a rationalized approach, to this end an Integrated Environment for Programming has been developed where each module's requirements can be satisfied [16]. The guiding principle consists of programming the behaviour of robots, both real and simulated. Both Java and C are used and also sequential and concurrent programming paradigms employed. The Integrated Environment is deployed (i) as a multi-robot simulation written as a Java application (ii) as a distributed multi-robot simulation using Java client-server architecture running on the Web and (iii) on minimal electronic and mechanical hardware. Robots have had a long history of use in CS education both in simulations and as physical machines [7]. From the perspective of a general programming tool, these systems are restricted in software flexibility [10,14] and are hardware expensive [13].

The most exciting part of our Environment is the Web deployment where we enable collaborative learning of programming within groups of students, these may be situated at any geographical location. This is a contribution to research into the development of Open and Distance Learning tools [6] where this Environment may be deployed as a tool to study the formation and maintenance of Open learning groups.

This paper details the operation of the Integrated Environment, the structure of the software and finally the pedagogy. Section 2 summarises the local implementation (details are found in [16]) both simulator and robot, and section 3 discusses the distributed implementation. The software architecture is briefly discussed in section 4 and pedagogy in section 5.

2. LOCAL IMPLEMENTATION

The simulator in its simplest *local* form consists of a robot world which contains several robots, trees, rocks and other items and which forms the right hand panel of the GUI (see Fig.1). Students program the behaviour of the robots in the left panel. Students program in Java, compile the code (using the built-in javac compiler), load the code into the robots and observe their behaviour and interactions with objects in the robot world. The world has continuous physical space and is not limited to the block world of other systems [15]. This method of working has two very positive consequences. First the students get a *visualization* of their running program, they see the effect of changes in their written code in an appealing way. Second the system is very responsive – effects of their code changes become *rapidly* apparent. In short, our students find this an efficient, engaging and fun way to learn programming.

The "Braitenberg" vehicle approach to programming is used, here the students must modify or write code to elicit specific robot behaviours (move around a lamp, clear up some rocks). The programming paradigm used with

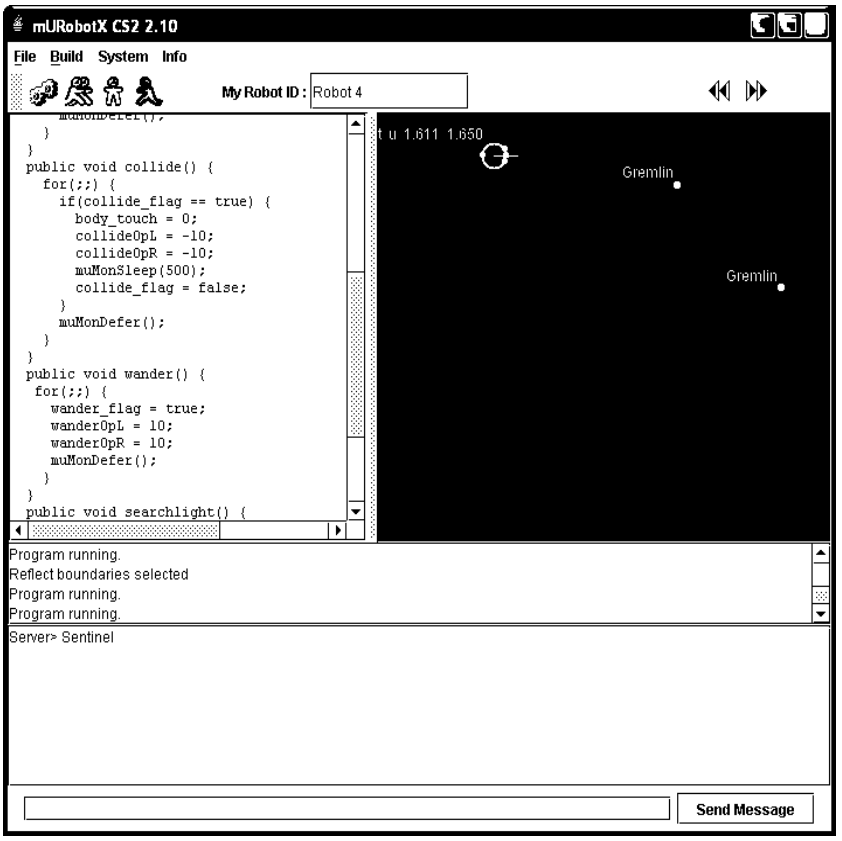


Fig.1 Screenshot showing Distributed Implementation of the Simulator with code window to the left and robot world to the right. Message area and Chat Room facility visible underneath.

beginning students is not sequential. It is concurrent! Following the "subsumption" architecture of Rod Brooks [4], a complex robot behaviour is decomposed, e.g., into "collide()", "wander()" and "searchlight()" blocks of code, each developed independently. A hidden scheduler will arbitrate between these blocks. Of course, when the system is deployed in Operating System class, the scheduler is revealed.

Indeed there are several versions of the *local* simulator available. One reveals the scheduler, another supports finite state machine (FSM) programming, and another allows the robot behaviour to be programmed via a back-propagation Neural Net. Of course it is possible to write purely sequential programs.

Even though the students are programming in Java, they are initially restricted to using a C subset of syntax. They are then able to cut and paste their programs into the physical robot integrated development environment (IDE) which is a professional product but freely available in a limited version [11].

3. DISTRIBUTED IMPLEMENTATION

The system has recently been re-engineered to allow multiple users to access one shared robot world on remote machines; this has opened up the facility for collaborative learning of programming. Here, each student programs the behaviour of a single robot which is then added to a common robot world (containing a mix of rocks, trees and other objects) which is normally selected by the tutor from a list of prepared worlds. Students can then see how the performance of their robot compares with their peers. The Environment is also fitted with a chat room where students can post and read messages. The student chooses an ID which is used for both chat room indexing and as a label personalizing each robot on the screen (see Fig.1). A further facility enables the student to add

his robot to the given world without any other robot being present. Typically the tutor will provide the group with a task (e.g., "produce behaviour to move towards then around any light source!"), the students will disengage from the robot world (while maintaining collaboration through the chat facility) and develop their code in their relatively uncluttered private world. When ready, they re-engage with the group world, and of course they discover the need to interact with a more complex environment. It is here they work on their "collide()" behaviour. This is one example of an emergent pattern of group collaboration, and in this sense our Environment can be used as a tool for investigating collaborative group work.

4. SOFTWARE ARCHITECTURE

There are several aspects to the software architecture. The overall design philosophy adopts natural (especially biological) systems as the design metaphor. This approach was taken to allow extension to modelling and simulation of natural systems. There is a hierarchy of levels in the architecture: At the bottom is the "Robot World". Robots, at the next level, exist as objects within this world and contain within themselves controller objects, the topmost level, where the students' code is located. The robot world is "runnable" and serves to mediate interactions, such as collisions, between the robots; it models physical space. Robots are endowed with sensors and effectors and maintain their state variables (e.g. position). All objects within the robot world, including trees, rocks, etc are extensions of a class "Thing". The robot world object, seen as the lower level in a hierarchy mediates sensory information, e.g. visual information, passing this up a level to the robots and other things. The robots will in turn pass this information up to the controller level. It's here that the visual information hits the student and is available to him as values of visual variables (values of the left and right eyes). In other words, the student programs the "controller". This upward flow of sensory information is mirrored by a downward flow of actuator information which defines the robot's movement, ultimately appearing as updated coordinates in the robot world. The student will program the controller to set the speeds of the robots left and right motors. Fig.2 summarises this aspect of the architecture.

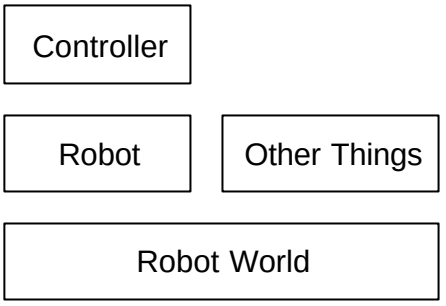


Fig.2 Software structure mirrors natural world. Bottom robot world contains robots and other thins (rocks, trees). Robots contain controller programmed by the student.

The robot activity is produced by a series of concurrent "behaviours". The student programs each behaviour as an independent method, such as "wander()" which is the base level behaviour, subsumed by "searchlight()" the behaviour executed when light is detected, subsumed by "collide()" which takes control when a collision occurs. Each of these behaviour methods runs as a separate thread of execution, which is scheduled by an "arbitrate()" thread according to a cooperative scheduling policy. This architecture allows direct transference of the student's code to the physical robot which runs a real-time operating system (RTOS) which has been designed to provide an isomorphic scheduler. The consequence of this architecture is that the students are programming in a *concurrent* paradigm rather than a sequential. Even CS1 students find this an easy and intuitive way to program. This approach is targeted at Operating Systems teaching where cooperative scheduling may be discussed, experienced and experimented with.

The environment can be configured to 'hide' parts of the code e.g. the scheduling methods are hidden when working with school children. This is made possible by our code compilation procedure: The source file is automatically appended to a hidden class source, and the union compiled using the Toolkit *javac*. The resulting class file is dynamically loaded at run time. This procedure is used each time the student makes a change to the code and hits the compile button.

Various design options were available when constructing the distributed implementation. The main consideration was the location of the runnable engine, whether to place this on a common server or at each client. A second and related consideration was whether to provide synchronous updates of one common dynamic robot world on all client machines, or else to broadcast the 'initial state' of the robot world and let each client produce a local and hence non-synchronous dynamic evolving world on its local runnable engine. Here however, each machine running at a different rate would cause divergence of the observed world. The chosen architecture uses local runnable engines coupled with asynchronous updating of all clients via a shared list which resides on a server, see Fig.3. This list comprises minimal "footprints" of all robot world objects, each footprint holding position, identification and state variables. During each iteration of the runnable client engine, each client requests the footprint list, uses this to compute interactions between the local robot and all objects on the list, and then updates the footprint of the local robot on the list which resides on the server. This has been found to be an efficient communication procedure. Of course, each robot runs effectively at the speed of the local client machine. Within a computer lab with machines of similar performance this is not an issue, yet if operated internationally some imbalances in observed robot speeds will occur. The chat room functions in a similar way passing text messages into and out of a common shared list. Java RMI is used to support this functionality.

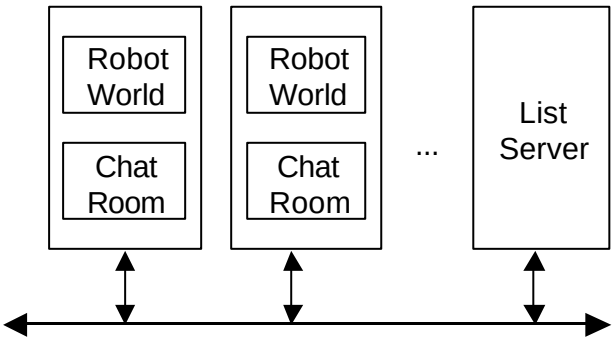


Fig.3 Distributed Architecture: Two students (left) exchange information via List Server. Computation done in local robot worlds, server only broadcasts robot states.

Each student runs a client application and programs just one robot. The tutor runs a modified client which contains the various robot worlds. This client initialises any static lists, but also contains a runnable engine so can support dynamic objects, currently these are light-emitting "gremlins". This architecture distributes the execution loading and communication needs efficiently across all collaborating machines.

5. PEDAGOGICAL ISSUES

How does this Environment fit into the CS curriculum? Leaving aside for the moment the Open and Distance aspects, the Environment can be progressively "exposed" to successive levels of classes. Currently, first year students see none of the underlying Java, they simply use the application as presented to learn to program. They effectively learn "C" syntax and are not introduced to object oriented programming (OOP). But second and third-year students are able to "roll back" the underlying code by loading the application into a Java IDE (JCreator, JBuilder) and inspect the various classes building up the application. Here they begin to learn OOP. A rich diversity of techniques has been used to develop the Environment including advanced techniques such as method reflection, dynamic loading of classes, integration of the javac compiler and the use of RMI and object serialization. There is adequate material here for level 3 students to inspect. On top of this inter-level progression, there is an intra-level progression learning methodology. For example, students meeting the Environment for the first time are asked to experiment with changing parameters in existing code blocks. This is a good way of motivating students, they get results fast, and also appreciate how the various code blocks work together. There then follows a discussion of concurrency Students are then asked to build sequences of actions (using the if - else construct). The notion of loops also emerges naturally.

In short, the students experience a robot world which directly reflects the effects of their actions, providing immediate feedback of their "what-if" experimentation, a hallmark of Constructivist learning. There is progression of learning which matches the cognitive development of individual students. This material is currently deployed *not* in

a "programming" class, but rather in a "digital technologies" module, originally as part of an internal learning and teaching project. Students taking both modules and who often find difficulty in synthesizing complete programs in the programming module report that the Environment, with its dual analytic-synthetic ethos, does facilitate their learning of programming. There is a real difference here; the "programming" students will often experiment to try to solve a given task (and this experimentation may be quite undirected), whereas this Environment guides students to experiment in a constructive and progressive way. Also, their experiments are *real* in the sense that they are carried out to solve (and understand) a problem grounded in the reality of interacting life-like animate creatures rather than an abstract problem of summing a series or sorting some data. In a student evaluation of the current semester's "digital technologies" module, when asked which was the most productive session, more than 50% of the 40-plus students indicated this robot activity. When asked which activity should be extended, again more than 50% indicated the robot activity.

Students indicate that they are highly motivated by the use of the Industrial strength IDE with the physical robots. Even though many do not spend significant time working with the physical robots, and even though some classes only experience a teacher demonstration, they unanimously report that the use of this IDE effectively validates the simulation work. The realisation that their work is relevant outside an educational (and pure "desktop") context seems important for them.

Concerning the open and distance aspects, Universities are responding to the increased student interest in distance learning with pilot projects which teach people how to collaborate in learning [1]. These projects are still young, it is too early to know which approaches and tools are useful. There is a lack of theory and still need for experimental studies. Current research has focussed on using synchronous activities over the internet such as simultaneous editing of documents [4], real-time meetings [2] and game-playing [17]. Software developers are also using collaborative tools such as email, file-sharing web whiteboards and interactive CAD software [5]. The University of North Texas has developed a useful environment for researching collaboration in software engineering [3]. This significant work provides many of the features mentioned above.

The conception of this Environment differs in many ways from these. First, it was not specifically and comprehensively designed to be a collaborative environment *ab initio*, indeed each stage in the development of this Environment was a response to students' observations and requests. Students organized themselves spontaneously into collaborative groups, sharing discoveries and facilitating learning. The faculty response was to re-engineer the Environment to provide a minimalist and easily modifiable collaborative learning facility. The current thinking is to deploy the Environment and observe how it is used by student groups in their learning and to look for *patterns* of collaborative behaviour. The students are clearly defining the agenda: It is expected that several iterations of refinement will be traversed in the coming period of development.

The nature of collaboration in the Environment is also novel. It is mediated through a combination of an active robot world, a private code-development area and a standard chat facility. This could be seen as a rich medium which supports cooperation in the sense of a real "shared information space", shown to be important in collaborative learning [8]. Ultimately, when the environment is stable, it may be appropriate to deploy it as a tool for investigating aspects of collaboration in Open and Distance Learning research.

This approach has been in place for one year and as reported above, student evaluation has been highly positive. The environment has also been successfully used with 14-16 year-olds and 16-18 year-olds within the "Aim-Higher" programme. While students have been constructively critical in the development of the Environment, it is perhaps disappointing to note that they have apparently missed one important issue. This concerns the *content* of their learning. While designed to teach programming, the Environment implicitly teaches about robots, systems and control and ultimately physics. It was not the intention to provide a systematic introduction to this material, and despite the engineering of the Environment to

avoid difficulties in this domain, they could naturally surface, and they did. Some students had difficulties with the details of sensing (such as inverse-square drop off of light intensity with distance), and indeed concentrated much effort on understanding (and in the main resolving) these problems. They were effectively learning physics, not programming at this point. Indeed, some students seemed to be learning programming "by osmosis", as a skill they had to master to understand a physical situation. In the context of the "digital technologies" module this was laudable, but clearly it may not be appropriate for a "programming" module. It does suggest that sometime in the future the most appropriate place to learn about physics and the other natural sciences may be through the programming of physical worlds (and the analysis of these worlds), rather than in the understanding of our own world.

6. CURRENT AND FUTURE EXTENSIONS

The general architecture may be described as being able to support collaborative learning with active objects, which are in this case robots. But these objects could be many other things, e.g. a collaborative Digital Electronics Workbench has been coded, where students can collaboratively drag-and-drop logic gates onto a shared breadboard, wire them up and simulate the logic. Deployed in the "digital technologies" module this allows rich learning of a technical topic. The intention is to encourage collaboration, but also facilitate teamwork where the task of building a relatively large electronic circuit composed of subsystems could be divided, each subsystem being produced by individuals or groups of students working together, then finally added to a common shared space where the entire team will observe the testing of the whole. Students trialling this Environment are already requesting enhancements to the vanilla Workbench, e.g. multiple design windows, which will be reported, in due course, elsewhere.

7. CONCLUSIONS

An Environment for learning programming which has several novel facets has been presented. It supports multiple language learning (C and Java) and several programming paradigms (concurrent, sequential, NNet, FSM etc). It supports both local and Open and Distance learning and provides an approach which can easily be extended and generalised. Its internal details can be rolled back to suit the cognitive development of the students. Students love it and remain involved in its evolution.

8. REFERENCES

[1] Atkins, D.E., Cogburn,D.L., Levenson,N., Mulvihil,M., Weilbut,M. Crafting Virtual Collaborations: Cross-National (US and South Africa) Learning Teams, Meeting Inter.St.Ass., l A CA, 2000

[2] S.Bly., S.Harrison, S.Irwin, "Measuring the effectiveness of robots in teaching computer science", *Communications of the ACM*. 36,1, 1993

[3] R.P.Brazile, K.M.Swigger, B.Harrington, B,Harrington, X.Peng, "*The Inernational Collaborative Environment (ICE)"*, http://zeus.csci.unt.edu/ 2002. [accessed Nov 2004]

[4] R. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation* vol RA-2 No.1, 1986

[5] E.Carmel, *Global Software Teams* NJ Prentice Hall, 1999.

[6] EDEN : European Distance and eLearning Network http://www.eden-online.org/eden.php [accessed 2004]

[7] B.S.Fagin, L.D.Merkle, T.W.Eggers, "Teaching Computer Science with Robotics Using Ada/Mindstorms 2.0" *Proceedings of the 2001 Annual ACM SIGAda International Conference on Ada,,* 2001

[8] I.Gorton, I Hawryskiewycz, L.Fung , "Enabling Software Shift Work with Groupware: A Case study", *Proc. 29th Conference on System Sciences"* Vol III 1996, IEEE Computer Society Press, Los Alamitos, 1996

[9] S.Greenberg, D.Marwood, "Media Spaces: Bringling people together in a video, audio, and computing environment.", *Proceedings Conference on Computer Supported Cooperative Work)* ACM New York, 1994.

[10] *Interactive C,* http://www.newtonlabs.com/ic/ [accessed Oct 2003]

[11] Keil Software athttp://www.keil.com [accessed Nov 2004]

[12] J.B.Knudsen, "*The Unofficial Guide to LEGO MINDSTORMS Robots*" O'Reilly, 1999

[13] *K-Team: Khepera II Mini Robot,* http//www.k-team.com/robots/ kephera/ [accessed Oct 2003]

[14] Lego Mindstorms, http://mindstorms.lego.com/eng/default.asp [accessed Oct 2003]

[15] R.E.Pattis, "*Karel the Robot: A gentle Introduction to the Art of Programming"*, Wiley 1995

[16] C.B.Price, "An Integrated Programming Environment for Undergraduate Computing Courses." *Proc. IT Research and Education Conference 2004 LMU England*, 2004

[17] S.Turkle, "Life on the Screen: Identity on the Age of the Internet", Simon and Schuster New York, 1995

Java and BlueJ – a good blend?

Gillian Rawlings
Department of Computing & Information Systems
Edge Hill College
Ormskirk
Lancashire

Abstract

This paper presents our current approach to teaching object-oriented software engineering within our department. It presents details of the initial choice of language and platform, the implementation of the module, the students' experience and results, and draws conclusions about the suitability of such an approach for inexperienced developers.

1. Introduction

Modelling information systems is a key stage and critical foundation to the development of information systems, irrespective of the methodology employed. The designers' goal is to produce a model or representation of an entity that will later be built [5]. The derivation and application of a model captures a holistic perspective of a system; the relationship between its objects, its semantics or its dynamic state, and a representation of its processes.

Modelling is a fundamental conceptual activity, and our second year module **INF2030 Object-Oriented Software Engineering** develops students' skills in this area.

Object-Oriented modelling is now a well-developed methodology; the Unified Modelling Language (UML) has also provided a common notation that is very widely used [6]. Many CASE tools support UML. The high costs incurred in software maintenance and failed systems have focussed attention on the importance of using adequate and appropriate modelling techniques. Large organisations and software houses will therefore seek evidence of some experience in information modelling from graduates employed to undertake such tasks.

The inclusion in this module of a programming language that supports OOP enables students to turn the abstract model into a software product, thus providing a coherent experience of the OO paradigm. This approach clearly links theory with practice.

2. Why use BlueJ?

There comes a point in the module where it is necessary to start looking at the implementation of a design. But which is the best way, and in what language?

Java is a relatively young object-oriented language. It has taken the software world by storm due to its close ties with the Internet and Web browsers [4]. It

is designed as a portable language that can run on any web-enabled computer via that computer's Web browser. As such, it offers great promise as the standard Internet and Intranet programming language.

Over the past few years Java has become more and more popular, especially as the chosen language to use when teaching OOP. But to use the full blown commercial IDE is very difficult because more time is spent getting to grips with the software then you do actually programming [1].

So why BlueJ?

My colleague posted this question on JISCmail in February 2002 [2]:

We are introducing Java as a programming language to 2nd year Information Systems students. They will be using Windows 2000 platform.
Can colleagues please recommend a good Java IDE? We are also looking to purchase a CASE tool to be used in conjunction with Java. Have you any recommendations?
We have used Poseidon this year; students found it extremely slow (even on P4 2GHz machines) and experienced numerous crashes.
Students have already experience of Visual Basic from year 1 (though it is likely we may move to Delphi - any comments welcome).

Chris Beaumont

There were 35 replies. There were suggestions to use JCreator and JBuilder, and those that said not to use either of these (overwhelming; less suitable for novice learner; difficulty in networking university edition), there were the odd suggestions to use IDEs developed especially for particular universities, but there were 8 institutions that recommended BlueJ.

University of Newcastle
University of Kent
University of Glamorgan
University of Keele
University of Lincoln
University of East London
Birbeck College, University of London
University of Leeds
University of Hertfordshire

All were in agreement that BlueJ was specifically aimed at beginners, gave a very good visual presentation and was easy to use. After further research a decision was reached – BlueJ seemed the ideal choice for students with no previous experience of programming in an O-O language.

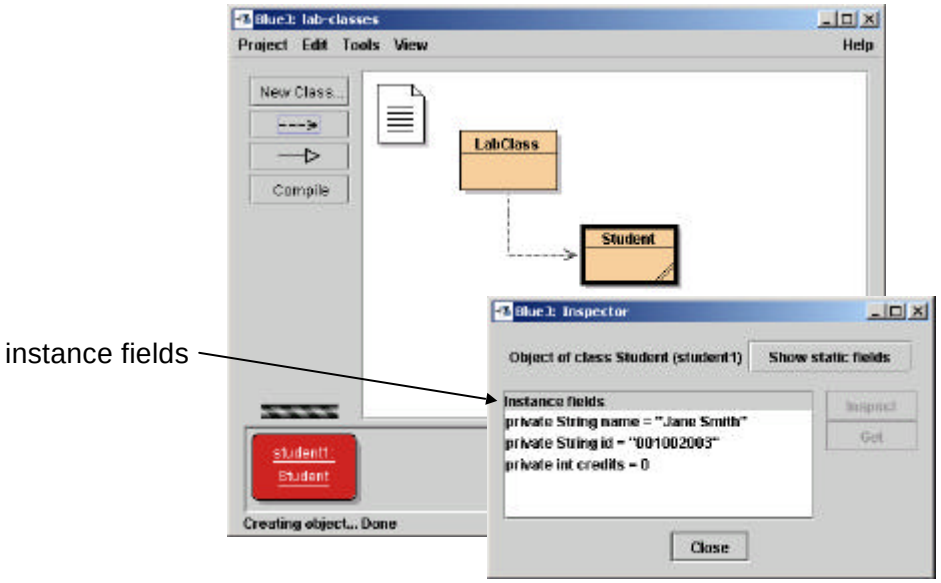
BlueJ is an interactive development environment with a mission: it is designed to be used by students who are learning how to program. It was designed by instructors who have been in the classroom facing this problem every day.
James Gosling, Sun Microsystems [1]

3. BlueJ

The book [1] seemed to be aimed clearly at a classroom audience who want an introduction to Java and/or an introduction to object oriented programming. It aims to teach both the language and the principles that underpin the language. The first and more obvious thing about the book is that it is supplied with a copy of the BlueJ interactive programming environment designed to help teach Java. BlueJ presents a clean development environment that allows the building, running and debugging of Java code. However it's not simply a Java IDE, it's a workbench that enables the user to instantiate classes, run methods, interrogate fields etc. In other words objects become real entities which can be manipulated and examined interactively in the workspace rather than abstract data structures that come and go while a program is running. We learn best by doing, and BlueJ enables the user to 'do'.

The book itself is structured around a series of topics (organised into chapters) and it develops the students understanding of BlueJ gradually, with plenty of practical examples. It encourages the students to try code out, to pull objects and code apart to see how things fit. The examples in the book are excellent. They are inventive, interesting and are used to really illustrate the ins and outs of object development.
The focus is clearly on learning about objects rather than on learning the finer details of Java syntax and coding styles. From the word go the emphasis is on objects and how they interact with each other. What is more there is also an emphasis on best practice. Throughout the text the questions are asked, 'how can we improve this code', 'how can we make it more flexible' and so on.

The BlueJ interface is very similar to a simplified UML diagram.



Right-clicking on a class shows a list of the class methods (including main if it has been coded) and each can be executed separately. If a constructor is

used from this menu then a red object is created on the object bench at the bottom of the screen. Right-clicking on an object shows a list of the instance methods (including those inherited from super classes), again these can be executed directly and will prompt for any parameters required.
Design principles are not neglected, with discussion of iterative techniques, design patterns and so on. Again the emphasis is on object development not just Java coding. Crafting solid, loosely coupled code is what's important rather than covering every single aspect of the language syntax and extensive libraries.

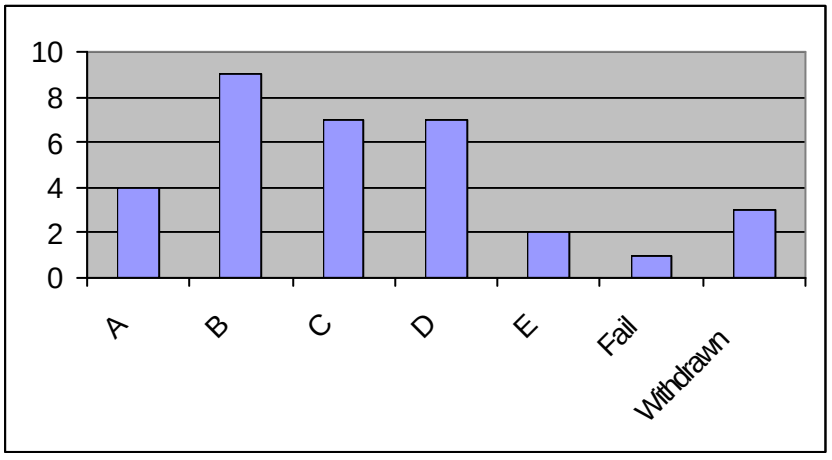
Advantages of using BlueJ are:

- BlueJ uses its own Java compiler. This supports an enhanced syntax error reporting system that gives access to more helpful error messages than those supplied by a standard Java compiler.
- Sample projects that are used in each chapter (on CD).
- Exercises that take the student from initially loading a project and running, it to making simple amendments and in later chapters to creating applications.
- It does not claim to cover all aspects of the Java language, but it's main focus is to convey o-o programming principles in general, not Java language details in particular.
- BlueJ is supported by an extensive web site with a tutorial, reference manual, links to other users and an active discussion list.
- BlueJ is free.

4. Student experience

The module runs on a fairly traditional presentation method of formal lectures supported by smaller tutorial and practical sessions. All material presented is available online, either locally on the college intranet or remotely via a WWW interface to a Virtual Learning Environment (WebCT), and can be accessed at any time. We do not have enough time to complete every chapter in the book, but cover chapters 1-6 thoroughly, and also spend a fair amount of time on testing. The first cohort (2003-4) used the first edition of the book [1]. This had one drawback – there weren't any GUIs. But this has been included in the second edition- so the current cohort will also be covering this chapter as well.

The module is not compulsory, and the first time it ran it only had 33 students enrolled. But this year we have increased that number to 45.



Results 2003-4 cohort

The feedback was varied, but mostly positive.

The assessment strategy was one initial piece of course work (in two parts) that was based solely on identify classes and drawing the associated Class Association Diagram using UML notation, a simple programming assignment, a more complex programming problem and an exam. The WebCT site contained various short tutorial exercises that were not assessed, and some example exam questions and quizzes. The students were required to show they used the discussion board in WebCT by including at least two postings and their associated threads as part of their first assignment.

Because the Java environment will run on any PC, the students did not have to rely on using the computers in college, but could do their assignment work (and any exercises they wished to re-do or complete) at home. Weak students were still failing to grasp the basic skills, but we added an extra lab slot on Wednesday afternoons where they could get more help. Stronger students were able to develop their skills by completing the 'extra' sections at the end of each chapter, which set them more challenging tasks.

There have been distinct advantages:

- Clear structure.
- Fewer problems with students getting started.
- Students are actively engaging in the exercises.
- Easy to review progress, which has enabled weaker students to be targeted for additional help.
- No operational issues.
- Software can be used at home without any problems.

And some issues:

- Not all students bought the book [1].
- Students found that terminology from other methodologies confused the issue e.g. entities, fields, attributes, associations, relationships

5. Conclusion

So is Java and BlueJ a good blend? Most definitely. BlueJ presents a clean development environment that allows the building, running and debugging of Java code. However it's not simply a Java IDE, it's a workbench that enables the user to instantiate classes, run methods, interrogate fields etc. In other words objects become real entities, which can be manipulated and examined interactively in the workspace rather than abstract data structures that come and go while a program is running. We learn best by doing, and BlueJ enables the user to 'do'.

The ideal student development environment needs to be simple to learn, easy to use, support the edit/compile/run cycle effectively. BlueJ offers this type of learning environment. If Java was the only programming language the students had to learn, and BlueJ the only platform, then we would have won the battle!

But it isn't, and this is where we loose the war.

Coard & Yourdan [3] argue that object-oriented techniques and structured techniques are compatible, and that the best ideas of both techniques can be used together. This is definitely not the experience so far with students. They get very confused when trying to understand the 'new' concept of O-O, especially when they seem to be so very fond of structured methods. We have yet to see a third year software development project that doesn't follow a structured analysis and design methodology, and isn't implemented using a database!

References

[1] Barnes & Kölling (2003*), Objects First with Java A Practical Introduction using BlueJ,* **Prentice** Hall / Pearson Education

[2] Beaumont C, LTSN-ICS-JAVA@JISCmail.ac.uk, posted 4/02/2003

[3] Coad & Yourdon (1991), *Object-Oriented Analysis*, Prentice Hall

[4] Lervik & Havdal (2002), *Java the UML way*, Wiley

[5] Pressman (2005), *Software Engineering –A Practitioners Approach*, McGraw-Hill

[6] Stevens & Pooley (2000), *Using UML: Software Engineering with Objects and Components*, Updated Version, Edition Addison Wesley

Selecting Students for First Diploma or National Diploma in Computing based on E-assessor.co.uk Diagnostics

James Vickers

Boston College, Lincolnshire

Abstract

Educational research already exists in the field of programming abilities and aptitude in an attempt to prove the link exists. The result of this research which analyses further education students does indeed show a correlation between aptitude and computer programming.

The results of this paper shows that from a significant analysis of learners and their associated skills, the test outlined in this paper is able to show both aptitude for programming, and evidence of whether students abilities within this area have been developed from learning programming or not. The aim of this is to therefore improve retention and achievement.

1: Introduction

Computer Programming has been an area which has been researched within the Computing discipline for many years, mainly because of the difficulties that many individuals have with learning the key concepts associated with the basics of computer programming.

Computer programming, as a science appears to have many commonalities with other science based subjects, in that it requires a number of abilities in a learner of the subject. The main abilities that become apparent are that of problem solving and logical thinking, combined with basic mathematics and the ability to follow a structured approach to analysis.

The modern computing courses are multi-disciplinary, allowing students to study whatever field of computing they find the most interesting, but unfortunately, this does not distract from the fact that many students at the outset desire the glamour and status of subjects such as computer games programming. This has mainly occurred because of the advent of the home computer market.

Therefore a major difficulty that most students encounter on computing courses are the programming units, which to them are the attraction factor to the course, but tend to prey most on their minds, and on their time.

2 Issues around programming based courses

2.1: General Computing Issues

The main issue with computer science courses is that in the majority, they require the student to undertake at least one module / unit that covers the basics of computer programming. Jenkins (2002) quotes that *"Few computing educators of any experience would argue that students find learning to program easy"*.

As discussed by Jenkins (2002) research has been undertaken in an attempt to provide solutions to the issues of how to teach programming, but teaching programming to someone who is eminently not suitable for the course becomes frustrating, and as discussed by Allison et al (2002) leads to a high drop out rate.

Allison et al (2002) summarise their reasoning for analysing programming abilities due to:

> *"The teaching of the initial programming language is a significant pedagogical problem for computing departments. With increasing numbers of computing students in the UK, the teaching of this core skill has become more difficult. A higher proportion of students are selecting to study computing without a natural aptitude for programming."*

The research within this report is not to discredit research previously made, but rather to complement it by posing the question of analysis of students prior to the start of their course. The aim is to see if a mechanism can be devised that based on previous experiences, can show students what could be a more suitable course for them. The ideology of this is to create a diagnostic test that can improve the retention and achievement of students across a curriculum area that historically has suffered a high drop-out rate. Allison et al (2002) summarise their experiences at NTU as follows:

> *"The consequence of these changes is that a diverse set of students enters university to study computing [7]. Many of these students are not equipped with an appropriate problem-solving aptitude for learning programming. So, computing courses have wastage rates that are high compared to other disciplines."*

2.2: Further Education College Issues

For 16 years, Boston College has been offering the BTEC First Diploma in IT Applications and the BTEC National Diploma in Computing as their main stream options for students wishing to gain qualifications and careers in the IT or Computing industries. Many of the students, especially with the continued expansion of the home games market, have decided that Computer Programming seems the most interesting field to work in.

A further problem with the selection process has been the nature of the rural location of Boston College, and the associated financial status of many students background. The concept of a career in computing, still caries the kudos of a well paid job, earning many thousands of pounds – even though the industry has declined since this period.

The reality of this is the opposite, where the industry is becoming tougher to enter, and more diversified with companies wanting specialists in single subjects, or alternately, literate in all aspects of the computing domain. In Further Education, the realisation of this is difficult to transpose into the course, and even when effectively done it is rarely taken on board by the majority of students. The current job market for IT means that students leaving Computing and IT courses must now compete against more qualified individuals, and as such, they need to be effectively trained in a discipline that they can truly excel at. Nobody denies IT and Computing positions can be lucrative, but the financially rewarding positions are all the more difficult to achieve in the current market.

However, the stigma of the old job market seems forever attached to computing courses. This is supported by the work of Allison et al (2002) who discussed the similar reasoning for students applying to BSc courses at Nottingham Trent University were that the *"...IT skills shortage and resulting lucrative job market entice(d) many school leavers into undertaking computing degrees"*.

Previously, no assessment of student abilities in computing had been undertaken. The application procedure prior to the inception of the test outlined in section 3, was merely whether the student had achieved the correct number of GCSE grades or not.

3: The diagnostic system

3.1: Test analysis

Many different forms of computing courses exist, so the problem was posed to see if a system could be devised to help assist the selection process for students onto computing courses, almost as a careers advisory tool does.

From an analysis and requirements gathering exercise from 11 members of teaching staff at Boston College, the following issues were raised with students on computer programming courses:

- Students cannot grasp what a data type or variable is;
- Students lack of semantics and positioning abilities with statements is a concern;
- Basic sequence, selection and iteration cause the most problems with students;
- Students do not try different approaches to solve a problem;
- When students are given a sample of code, they cannot transpose the detail into an example of their own;
- An attention to detail, and a willingness to re-read theoretical materials is not evident;
- Students will not spend time reading questions carefully;

- Students do not take into consideration pre-requisite facts given to them;
- There is tendency for students not to bother modelling and conceptualise what is happening, or draw diagrams to figure it out themselves;
- At a basic mathematics level, students are unable to add simple amounts together iteratively;
- Students do not refer back to statements made to them in a question on exams;
- Students do not persist with problems, they merely ask for assistance to gain the answer, rather than learn for themselves

During the analysis, the 11 members of staff questioned tutored groups of students who are studying a range of 5 separate programming languages. All members agreed that these were issues of concern with the majority of students.

A diagnostic system was therefore devised to allow the staff to undertake an analysis of a student's ability to write computer programs. The test, which is administered using the multiple choice questioning website e-assessor.co.uk, presents students with a number of options which may be applicable to questions that are aimed to analyse their problem-solving, basic mathematics, reasoning and logical thinking abilities.

When the analysis was complete, it was the decision of the test author not to implement the test in a language specific sense, or expect prerequisite knowledge, rather it was decided to provide students will all of the information within the question. In this format, the student's abilities would be tested greater, as they would have to learn something first before answering the question.

3.2: Test format

The initial test devised in 2003 was 27 questions in length, and had weighted questions between 5 and 15 marks. The purpose of the weighting of marks was to allow for more difficult topics to carry a greater reward for the abilities required to solve them.

Two types of questions were used. Recollection of memory questions were introduced in an attempt to identify basic capabilities in iterative maths, and substitution of letters for numbers in an attempt to identify issues with conceptualisation and variables. The second type of question is based around reasoning, where the students are actually given the answer, but it is up to them to decipher the answer, or alternatively, choose the most applicable. It was hoped the second type of question would show their logic, and when the questions taught them so, their aptitude for basic programming.

A decision about the testing engine was to design it such that students could not review questions once they had been chosen. The main reason behind this is that computer programming students quite often make quick and unjustified conclusions without thinking about the consequence of what they have done. A second design characteristic of the test was to devise it using the capacity for a student to "pass" on a question. The reason for this was to identify the students who lacked persistence, and those that could not grasp the concepts of multiple choice, and having the chance of fortunately choosing the right answer.

The test automatically and immediately marks the test and reports the results to students, whilst logging this and making the data available to the staff members via their own menu system.

3.3: Differences to alternative tests

One of the biggest issues with the educational research so far, is that it focuses on students entering degree programs of study. This system has primarily been designed for use at levels 2 and 3, and not for levels 4 and 5 although it could be adapted at a later date. It is the belief of this report that the main reason for the failure of the aptitude system at levels 4 and 5 is two-fold.

Firstly, the majority of aptitude tests seen so far, such as the Huoman programming aptitude test (Monkkonen and Tukiainen, 2002) use terminology and semantics that students have not yet been taught, if it were indeed a pre-entry aptitude identification test. The test devised here does not use a base language, nor does it insist on the students writing pseudo-code. In fact one of the main test aims is to provide the students with facts at the beginning of the question before they are examined – in a mode to mimic the activities of any real teaching session.

Secondly, many of the tests are too focused on generalised problem solving, which is undoubtedly important, but not really what can be reliably tested on alone for an aptitude for programming. The test devised here incorporates all of the three major basic programming principles of sequence, selection and iteration, and combines these with debugging based questions using metaphors from the real world.

4: Results

Educational research into aptitude testing has produced inconclusive results as to whether or not aptitude testing is indeed a mechanism for testing students. Jenkins (2002) when referring to computer programming, quotes that:

> *"There is nothing inherently difficult in the subject; it is simply that some students have no aptitude. The skills often cited are problem solving and mathematical ability"*

If a historical look is cast upon the process of programming aptitude testing, many research papers have been aimed at the negative results of such a test, even though they contradict their work at times by acknowledging successful tests such as the IBM Programming Aptitude Test (IBM PAT) (Monkenen and Tukiainen, 2002) or the Berger Aptitude For Programming Test (B-APT) (Pyschometrics, 2004). This imbalance in decision by previous research does appear to provide a justification for further tests such as that defined in section 3, to be undertaken.

As discussed in section 1, the associated skills that many computer educators believe are required are those as defined by scientific subjects named in section 1. Jenkins (2002) quoting the work of Jenkins and Davy (1999) discusses that prior aptitude tests by the University of Leeds have proved that no significant results exist with students who have taken aptitude tests.

The results of this test have not proven inconclusive, but rather have shown some interesting patterns.

4.1: Comparison of aptitude and GCSE grades

The entry requirements for the courses analysed are that a student who enters the First Diploma in IT applications at Level 2 is required to have 4 D's at GCSE including Maths or English. A student who enters the National Diploma in Computing at Level 3 is required to have 5 C's at GCSE including Maths or English (with the other grade a D).

In the initial test year of 2003 (the first year the test has ran), all students entering a further education computing course at Boston College undertook the test. The test was initially conducted on a diverse base of the population which included academic staff, non-academic staff, programmers, non-programmers, level 2 and level 3 students, and non college adults who were not on any course at all. A decision was made that after trialling the test with a diverse base of users which, and by looking at their educational profiles and their associated scores, a boundary of 51% was decided for a level 3 and above educational level.

When all 51 students had taken the test, only 4 from 14 students who gained a place on the First Diploma due to their GCSE grades gained a score higher than 51%. A success rate of 71.42%

When all 51 students had taken the test, only 3 from 37 students who gained a place on the National Diploma due to their GCSE grades failed to gain the score of 51%. A success rate of 91.89%

Overall, from the 51 students tested, the test correctly analysed 44 students against the GCSE requirements deemed by both the awarding body and the institution. This proves a success rate of 87%.

The 7 anomaly students have since been analysed and questioned with permission. The results of this analysis are as follows.

The three students who did not meet the 51% requirement for the National Diploma:

- A mature student returning to study from a manual labour position in a food packaging warehouse

- A student progressing from First Diploma to National Diploma, who would have rather studied general computing and not software development
- A student entering the National Diploma who would have rather studied general computing and not software development

The four students who exceeded the 51% requirement for the First Diploma:

- A student who has been programming for 2 years before starting the course
- A student whose father works in the Computing industry, and had given him some brief training before starting the course
- A student who came from a school with serious staffing issues in English and ICT where he did not succeed in GCSE grades
- A student who suffered from serious personal and family problems, and as a result missed many weeks of school.

It has on many occasions been an issue with students who enter the courses defined above, not to have the pre-requisite skills identified by the GCSE system. Many cases have arisen on enrolment days, where students and their parents have argued furiously with admissions staff that they are more than capable of a higher level course. In many cases, the students may indeed be capable of one or more of the aspects of computing, but the purpose of the GCSE system is to identify a rounded individual and cannot be dismissed as an additional requirement. However if a student has not gained any GCSEs, such as the case with some mature applicants, or if students have had medical problems that have prevented their attendance at GCSE then there is a often a reason for the absence of GCSE grades. A question has always been placed by admissions staff at this level of how can they make informed decisions on students of this type? One interesting result from these initial results has shown that the test here can indicate an approximate level of attainment at GCSE, if other data is absent

Although the system is still in its infancy, it has proven an effective tool for identification of ability for those with little or no GCSE grades to view.

4.2: Comparison of aptitude and performance

All National Diploma in Computing students study the software development module in the first 11 weeks of their first year of study. To gain a comparison of the aptitude of students, all students software development combined phase test grades were compared against their initial diagnostic test score to see if any results could be concluded. The students performance data was analysed with an attitude and effort report from their lecturer as part of the Boston College parental report processing procedure.

The results showed that the top 5 students on the diagnostic test made minor improvements in their comparative software development overall grades. The 2 students who wanted to study general computing, who did not pass the diagnostic test did not find the unit easy at all, and scored less than their diagnostic test. The student who had been working previously, improved by 8% with a great deal of determination.

The remainder of students varied in their performance. The students who applied little effort and did not revise for the phase tests suffered the greatest decline in performance between -9% and -29%. Students who did revise for the phase tests and applied effort varied with –5% and +15%.

The results can be classified as thus:

- Initial students who did not want to the study the course did not perform well in the unit, and did struggle through a lack of interest and aptitude
- Students who initially successfully passed the test but did not apply themselves in the software development unit did not perform as well.
- Students who had studied a previous programming language on a pre-requisite course suffered a small negative deviation in grades due to semantic differences in the languages.
- Students who applied themselves and revised for exams saw an increase in percentage on the unit.

4.3: Resit test

After the completion of the software development module, the students were asked to resit the diagnostic test to see if they had been up-skilled in problem solving and reasoning, by being taught the

principles of programming. The purpose of the resit was to validate and prove that if students are taught computer programming then there should be an increase in their diagnostic test scores as their abilities to write computer programs improve. If there is indeed a link between the scores on the diagnostic test and their abilities to write computer programs, then regardless of their performance on the software development unit, there should be a visible improvement in their abilities tested by the diagnostic system.

A total of 25 out of the 37 students took part in the optional resit test to see if they had been up-skilled by the process of the tutoring of the software development course. In the results, 21 out of the 25 students showed an improvement in their results by between 1% and 25%. This was also on a slightly different test where the high scores were harder to gain due to the weighting of questions being less generous. The four students whose scores decreased by between -3% and -6% had already scored exceptionally high the first time around, and probably suffered the deviation by the removal of such high weighting on some questions. 3 out of 4 of these students had gained scores in the top 50% the first time the test was sat, and were still in the top 50% in the second resit test.

The average improvement made by a student was 7% on their entrance mark, with one student gaining an increase by 25%. Another important result is that the students at the lower and upper ends of the initial test, received the highest improvements in results showing excellent differentiation of abilities taught by staff.

5: Conclusion

5.1: Conclusion of results

Although negativity has been applied in the past by previous research to the concept of proving that aptitude can relate to computer programming, and in turn this can evidence issues with students and GCSE studies, the results of this paper cannot be easily dismissed.

What the findings of this paper show, are that aptitude and programming do have a link as evidenced in section 4. The links between aptitude and GCSE attainment links can also be evidenced for students who wish to participate in more a vocational subject that their learning styles suit, rather than academic subjects that their learning styles do not suit.

The results of the tests show that teaching programming do indeed improve the abilities of students to solve problems and think logically, and thus evidence a link between the associated skills that some previous research had dismissed.

One main factor in the success of the diagnostic test and the associated results of many of the students has been their willingness to succeed, and their determination in the subject. This is especially evident in the students who have significantly improved both their diagnostic test score, and have gained very impressive phase test scores in the real test of their coursework in software development.

5.2: Ethical application

It is important to note that the purpose of this diagnostic test is to act as a careers advisory tool, and not as mechanism for preventing or limiting students access to courses. The main purpose is to allow an informed advisor to indicate to a student what the reality of a course entails, and what experience has shown them to be the norm.

An ideal application for the diagnostic test would be to inform a student who achieved a poor result the further areas of computing and IT that they may alternatively study, rather than just software development. It is also a useful tool for an advisor who knows little about software development, to illustrate to a student quite how difficult the subject is.

A further application is as simulation tool to predict equivalent, or current level of abilities within the range of GCSE grades (as discussed in 4.1). The use of this tool then expands to adult learners of computing that require an indication of their current level of ability, and as such, can gauge what course they should undertake based on their score.

5.3: Future Developments

The test question base is now to be expanded to test on multiple occasions all of the skills deemed necessary in section 3.1. The test system, as discussed previously in this paper, has been made slightly more difficult in an attempt to concentrate the results of the students into a more uniformed pattern by removing the weighting of many questions. This has been done in an attempt to reduce high scores being accidentally achieved by fortunately selecting the correct answers.

It is the opinion of the department that uses this test to pilot the scores attained by students as supplemental to their GCSE grades at enrolment for the 2004/2005 entrance to the courses.

Therefore, it is the conclusion of this paper that aptitude can be measured and used as an indicator toward identifying the abilities of students to write computer programs, but aptitude is worth nothing without determination and a willingness to learn.

6: Acknowledgements

Acknowledgements are made to Mr Stuart Hardy (Lecturer in Computing), Mr Doug Eke (Course Co-ordinator for the National Diploma), Mrs Vanessa Martin Stevens (Course Co-ordinator for the First Diploma) and Mr Mick Bean (Programme Leader for Computing) from Boston College for assistance in administering diagnostic tests.

7: References

Allison. I. Orton. P. Powell. H. 2002. A Virtual Learning Environment for Introductory Programming. In Proc 3rd Annual Conference of the LTSN Centre for Information & Computer Sciences. Loughborough, UK. pp 65-71.

Decker. A. 2003. I Want to be a Computer Scientist When I Grow Up: Evaluating the Skills Necessary for Computer Science. In Proc SIGCSE DC Conference. [Online]. Available from: http://www.radford.edu/~sigcse/DC03/participants/decker.html. [Accessed: 21st April 2004]

Huggard. M. 2004. Programming Trauma: Can it be avoided. In Proc Grand Challenges in Computing Conference: Tyneside, UK [Online]. Available from: http://www.cis.strath.ac.uk/external/educ_grand_challenges/docs/Huggard%20M.pdf [Accessed: 21st April 2004].

Jenkins. T. 2002. On the Difficulty of Learning to Progam. In Proc 3rd Annual Conference of the LTSN Centre for Information & Computer Sciences, Loughborough, UK. pp 65-71

Monkkonen. E. and Tukiainen. M 2002. Programming Aptitude testing as a prediction of learning to program. In Proc PPIG 14, Brunel, UK pp45-57.

Pyschometrics Inc. 2004. Berger Aptitude for Programming test (B-APT). [Online]. Available from: http://www.psy-test.com/bapt.html. [Accessed: 12th February 2004]

James Vickers,
Higher Education Co-ordinator for Computing,
Boston College,
Skirbeck Road,
Boston,
Lincolnshire,
PE21 6JF.

Teaching Information Systems Development with SMS

Chris Wallace
Information Systems School
Faculty of Computing, Engineering and Mathematical Sciences
University of the West of England
Frenchay, Bristol BS16 1QY

chris.wallace@uwe.ac.uk

Abstract

'Texting' or SMS mobile phone messaging is rapidly increasing in business and community use. This paper discusses the inclusion of SMS technology in the teaching of Information Systems Development. It is argued that SMS has advantages in terms of simplicity of development, encouragement of good development practice and the breadth of information systems concerns exposed.

Key words

Texting, SMS, mobile communication, information systems development.

1. Introduction

The final year module *Information Systems Development 3* (ISD3) is taken mainly by students on our Computing and Information Systems award. These students have had some exposure to Java in their first year and to Microsoft Access application development in their second year. In the final year we seek to broaden the range of application technologies by introducing 3-tier web-based applications. Typically these have a browser in the presentation layer, PL/SQL or PHP in the application layer and Oracle or MySQL in the persistence layer.

One difficulty with this module is to get a balance between the concerns and approaches to Information Systems Development on one hand and the narrower concerns of program and database development on the other. The goal here is not to develop great programming or software engineering ability but to develop awareness of information systems in their organisational, human and societal context, addressing issues of the role of information itself, its meaning, quality and value, human-human as well as human-machine communication and the design of business process.

Browser-fronted applications, as a technology to support information systems, are full of interest and possibilities but in the opinion of the author, they also present some pedagogic difficulties. These include the complexity of development, a limited range of communication possibilities and complex implementation for stateful interaction. This year we have turned to mobile phone applications in an experiment to increase the focus on information systems issues whilst continuing to engage the students' attention.

2. Texting and its applications

'Texting' or SMS (Short Message Service) is pervasive and ubiquitous. Figures of the volume of text messages sent world wide and in the UK are staggering. A survey of students on this final year module show 100% mobile phone ownership of which 100% use their phones for texting. Among students, SMS is largely used for personal communication. More advanced technologies such as MMS and WAP are as yet little used and few can afford high end Smartphones.

As an application technology, SMS is increasingly incorporated into organisational information systems - for marketing and fund-raising, for communication among staff, clients and members of the wider organisation [Cheverest el (2003)] as well as for on-demand information services. Within education, SMS is used for administrative support, for example to communicate alerts of closure to parents and exam results to students. SMS is also in use in direct support of learning through incorporation in e-learning systems [Soon and Sugden (2003), Attewell and Savill-Smith (2004)]. SMS in entertainment extends from voting on Pop Idol to public message displays in clubs.

We believe that the rapidly increasing use of SMS in information systems on the one hand and the relative simplicity with which SMS systems can be developed on the other, provide the opportunity to

incorporate this technology into the Information Systems curriculum with benefits to the teaching of Information Systems concerns and to the student's employability.

3. Overview of SMS Technology

SMS is a complex service within GSM. [Peersman et al (2000)] Here we are only concerned with the end-user view of the service and the three varieties of SMS service: out-bound, 2-way and conversational.

3.1 Out-bound

An out-bound service allows an application to send text messages to one or more known mobile phones. Such messages are termed 'mobile terminating' (MT). With the assistance of an SMS service provider who provide the channel between the application and the mobile phone network, this service is simple to establish. Typically the application program issues an HTTP GET request, with the number, message and other details encoded in the URL. The service provider then hands this off into the GSM network for delivery. Delivery notification can be returned to the application via a call-back URL. In additional to simple text messages, monophonic ringtones and operator logos can also be sent. Setup is typically free and the cost per message of about 4p is significantly less than the networks change a mobile phone user to send a message.

An out-bound service is good for 'pushing' information to subscribers – for example in schools for sending notification of an absent child to its parent or sending traffic information alerts to drivers.

Within the CEMS faculty, students can enrol in a notification service which will text alerts about class cancellation.

3.2 2-way SMS

2-way SMS allows a user with a mobile phone to send a text message which is received by an application, which can then reply with a message back to the originator. A 2-way SMS service requires a GSM number to which users can text a message. The message is routed through the service provider to the nominated user's application script, again using HTTP. Such messages are termed 'mobile originating' (MO). The application will then perform some task such as a database look up and then may text back a response to the originator.

A 2-way service is more costly since it requires the rental of a number and a small charge for each incoming MO message on top of charges for any outgoing MT messages. From Clickatell, who are our SMS bulk service provider, the current charges for a standard number (i.e. not a short code or premium rate number) are roughly £60 for setup, £15 per month rental and about 1p per incoming message.

Bi-directional communication with 2-way SMS allows the development of more interesting applications. For example at UWE, we have developed a service which allows students and staff on the campus to find out the times of the next two departures on any bus service leaving the campus. In future, users will be able to request a reminder about a selected bus departure. We have also used our 2-way service to implement a poll on the proposed ban on smoking in pubs and clubs (supported by 2 to 1) and on the US presidential race in November. Sadly this poll did not predict the real outcome!

3.3 Conversational

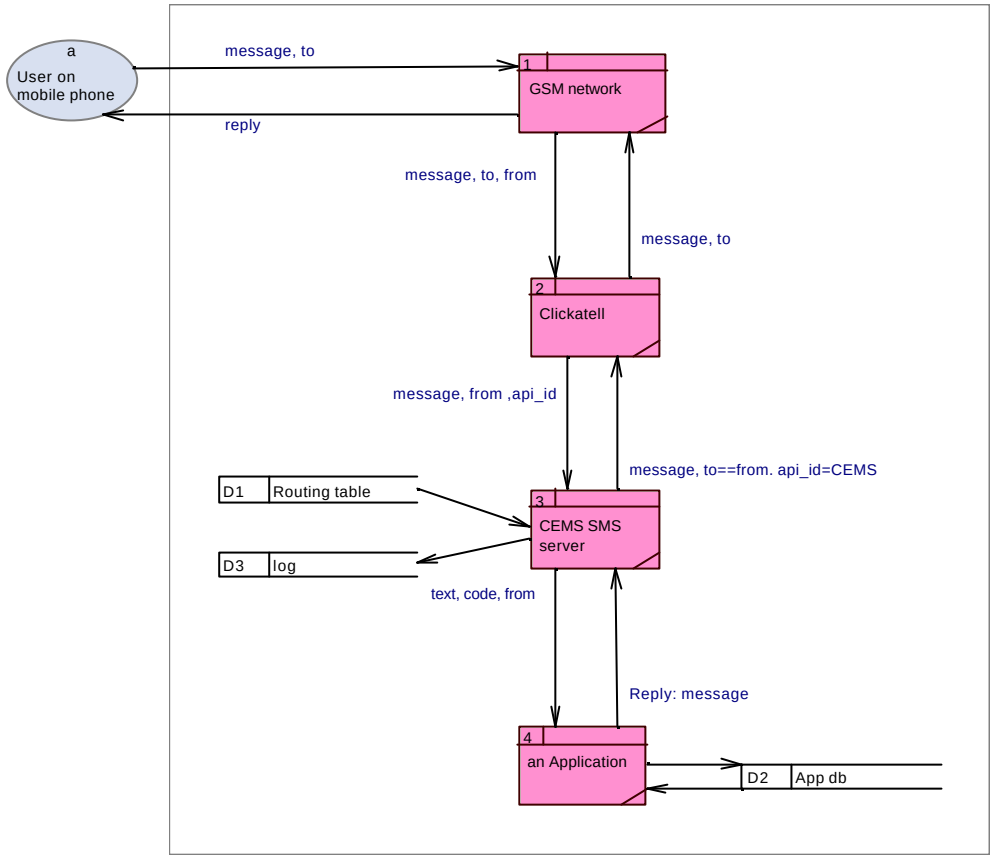
Ideally a mobile user should be able to reply to a message sent from an application to maintain a conversational interaction. By default, MT messages sent via a service provider appear to originate from that provider. In our case, users will see a +27 number since Clickatell is based in South Africa. We have yet to set up a conversational service due to restrictions placed by U.K service providers.

4 CEMS SMS server

For use as a teaching and project resource with the Information Systems school, it must be easy for applications written by students in coursework or in projects to use the common rented GSM number, to be able to generate a reply and for logging to be performed. A simple SMS server to handle these functions was written in PHP. Incoming messages received from the SMS service provider are distributed to registered applications, again using HTTP, on the basis of the initial word in the message. The reply from the application, if any, is returned through our proxy server and the SMS service provider to the originator. The SMS server also logs incoming and outgoing messages.

The level 1 DFD below summarises the data flows involved.

System (Level 1 Diagram)



Applications themselves can be written in any server-side language. At present we use PHP for its ease of interfacing with a web-server and database support. A new application is first tested standalone from the address line in a browser, from a test form in a browser, or from a test script. Once tested, the new application can be linked into the SMS server and the new service becomes immediately available from any mobile phone.

5. Advantages of SMS for teaching

5.1 Simple user interface

The user interface is very simple – a string in, a string out. This means that a simple application, such fetching the current time, can be written with only a few lines of code. Little more is required to compute the time in a given country using a simple database. The students' attention and potentially their creative skills can be focused on the human interface to the application and the specification of the functionality required. In contrast, browser applications tend to focus attention on the much more complex web interface.

5.2 Simple application interface

Similarly the application interface supported by our SMS server is described by three input parameters and a formatted output string. Such a simple interface allows the students to see the possibility for writing a functional specification, perhaps using a model-based specification approach, before writing the code.

For software development, an advantage of this simplicity is that it eases the use of a unit testing framework to automate regression testing. On this module we use PHP scripts for testing. The ease with which URLs can be treated as simply a filename simplifies the coding of test scripts. By contrast a browser-based application is much more difficult to place in a test framework.

5.3 Varied communication patterns

Browser applications provide a pull mechanism in which users request information from an information source. SMS on the other hand provides both pull communication via an MO message sent by a mobile user and push communication with MT messages sent from an application on stimulus from an operator, indirectly by another user or from a clock or other trigger. SMS can be coupled with email, HTTP and web service applications to produce a wide variety of communication patterns.

5.4 Simple stateful implementation

Stateless applications are the basis of web applications. Each request from a browser is processed without reference to any preceding requests from the same user. This limits the conversation between user and application since the application can neither reference nor develop a model of the user. In contrast a *stateful* application is able to recognise successive messages as coming from the same user and access and develop its own user model, the state of the interaction, with which to mediate its responses to messages.

There are various stateful mechanisms available to a web developer, differing by the location of the state, the lifespan of the state, the mechanism for user-to-state matching, the amount and structure of the state and its security and accessibility to other applications. Web mechanisms include hidden fields in forms, cookies, session variables and user models in a database. Each is widely used, but none are trivial to implement.

Since every message in SMS is accompanied by the number of the originating phone, there is a simple key available for user-to-state matching. Of course matching is not precisely one-one: people have more than one phone, and phones are lent and sold, but the alignment is closer than that provided by cookies which match to a specific computer login. Thus a simple database keyed by phone number opens up simple stateful applications supporting long-running transactions and business processes.

5.5 Challenging user context

SMS applications are nonetheless challenging for a developer. A user of our bus information service waiting at a bus stop will not have access to a web-page defining the message structure, nor to pull-down menus of choices available. Thus the information system designer must consider the use of their application by a mobile user in a variety of use situations. This focuses attention on what and where information about the service is available, on designing a message structure which is intuitive and on designing the application to guess at the needs of the user even if the message is not an exact match to the required format. Of course it is not always safe to act on a guess, for example in a share-trading application or in a control application. This raises the issue of the design of an appropriate human-machine protocol to suite the context.

These applications pose a variety of interesting software design problems. Interpretation of an incoming message which may be ill-structured and roughly keyed requiring parsing techniques using regular expressions, closest match algorithms or a syntax-driven parser. Since the SMS server logs all incoming and outgoing messages, a developer can view the history to observe actual application usage. This encourages an adaptive approach to design based on evidence from actual usage. In addition, the simple interface allows interfaces to be quickly simulated with naïve users either on paper or by using a web-based mobile simulator.

5.6 Real world implementation

Applications such as the bus information system and the straw poll have now been ‘released into the wild’. We as developers will have to respond to the demands and unanticipated consequences of actual use. We plan to release selected work by students on this module at least for use on open days with prospective applicants for our awards. We think it important that issues which arise from the use of new technologies are experienced in the raw and not merely treated as laboratory exercises. Functionally simple but pervasive SMS applications seem ideal for this purpose.

6 Concerns

There are naturally some concerns over the use of mobile phones in teaching information system. These include issues of security, user acceptance and programming difficulty.

6.1 Security

If desired by an application, additional security can be provided by requiring an initial password together with a timeout. If there is concern about the privacy of the numbers themselves, mobile numbers can be encrypted. However it must be noted that typically the service provider and others in the chain between application and mobile will log messages and the SMS channel is easily monitored by the Government. Without encryption of SMS messages on the phone itself, messages will always be insecure.

6.2 User acceptance

Mobile phones are almost exclusively used for personal communication and their use as a component in an information system can seem intrusive. Choice of application here is important – although an application to get the current time is easy to write, no one will spend 12p to use it.

To address this issue, we are developing a mobile network simulator which will allow information systems incorporating SMS to be exercised in the laboratory without the overhead or inconvenience of phone usage.

The cost of the service can be used to advantage however: it focuses attention on the value of information. If the value to a given user does not exceed the cost to the user, it will not be used. Our SMS system does not in fact provide a very realistic context since we pay for out-bound messages and information is not charged at market rates..

6.3 Programming difficulty

Writing an application to provide a simple response to a stimulus requires a minimum of knowledge of the chosen scripting language, but more complex applications stretch the students' ability. Whilst sample scripts for more complex functionality, such as database access, email generation or use of a web service can be supplied by teachers, these can still be hard to adapt for a new application. Given that our goal is to focus on the information systems we are creating rather than on programming, we acknowledge that we still have work to do to enable those less interested or gifted in programming to gain fully from this approach.

7 SMS in Information Systems teaching

This is the first year in which this technology has been taught and thus any observations are tentative and based on experience in teaching on the ISD3 module. Preparatory tutorials on scripting languages (PHP) and an SQL database (MySQL) were followed by a mini-project using SMS.

7.1 SMS project

The ISD3 module is unusual in having no assessed coursework and thus all exercises set are formative and voluntary. Additional motivation is generated by small prizes for the best submissions. The first SMS application which students were required to develop was a currency converter. For example, a user would text:

CONV EUR 100 USD

and receive a message

100 [EUR] = 127.89 [USD]

An initial tutorial explored the design and development of this project, exploring scenarios in which the service might be used, the design of the 3 tier architecture, the allocation of responsibility between the tiers, the design of the database to support conversion, the choice of ISO standard currency codes and sources of exchange rates and their quality.

In the next 55 minute workshop, students worked in small groups using a worksheet to create a simple database of currency codes and conversion rates (using Mark Dixon's QSEE tool), generate the SQL DDL and install in MySQL, install a supplied starter PHP script, amend the script to access the student's newly-installed database, and test via a supplied test form.

The final worksheet required the students, working in their groups, to enhance the basic application. Areas of improvement, gathered during the initial design tutorial, included greater flexibility in the format of the input message, better response to an invalid input, means for a user to bootstrap their understanding of the service, stateful interface so that the user does not need to re-enter the two currencies each time, and to provide a mechanism for updating the conversion rates. Students who

achieved any of these improvements then emailed the URL and a codeword for registration with the SMS server. They could then demonstrate their application to friends and family.

7.2 Results

By the end of the second session, the majority of students had a working copy of the basic system. A smaller proportion tackled the final worksheet and 5 groups (about 20% of the students) submitted applications. All had made useful enhancements even though their experience with PHP was minimal. One group had found a Web Service supplying conversion rates and created a script to update the database from this service.

Students generally enjoyed the work and some positively beamed when their application worked. In retrospect, more time and support should have been given to the exercise considering the number of new tools which were involved.

7.3 The Future

Students have so far responded well to the up-to-date feel of this subject and the number of students who want to undertake individual projects in this area has greatly increased. Further work is planned on our SMS server to provide better access to an application's traffic, a simulator for a mobile phone network within which SMS applications can be tested, and a kitset of parts for application assembly.

We are also working with colleagues in the Electronics and Computer Systems School to develop some SMS monitoring and control applications. Of personal interest to the author are yacht-borne systems which can alert a skipper on shore of problems aboard such water or human ingress, anchor dragging or changed weather conditions.

References

Attewell, J and Saville-Smith C (eds) (2004) *Learning with Mobile Devices: Research and Development,* Learning and Skills Development Agency

Cheverest, K et al (2003) , *SPAM on the Menu: the Practical use of Remote Messaging in Community Care,* ACM Conference on Universal Usability 2003, pp23-29

Clickall SMS Bulk SMS Gateway URL www.clicktell.com [accessed 7 Jan 2004]

Dixon, M., QSEE Multicase tool URL : www.qsee-technologies.com [accessed 7 Jan 2004]

Peersman, G et al (2000) , *A tutorial overview of the short message service within GSM,* Computing and Control Engineering Journal April 2000, pp79-89

Soon,l. and Sugden,D. (2003) *Engaging Students Through SMS Messaging*, in FERL Annual Conference 2003

Wallace, C., *SMS Services in the Information Systems School*. URL www.cems.uwe.ac.uk/~cjwallac/apps/sms [accessed 7 Jan 2004]